arXiv:2301.04237v3 [quant-ph] 11 May 2023

Solving the semidefinite relaxation of QUBOs in matrix multiplication time, and faster with a quantum computer

Brandon Augustino *†, Giacomo Nannicini ‡, Tamás Terlaky†, and Luis F. Zuluaga†

May 12, 2023

Abstract

Recent works on quantum algorithms for solving semidefinite optimization (SDO) problems have leveraged a quantum-mechanical interpretation of positive semidefinite matrices to develop methods that obtain quantum speedups with respect to the dimension n and number of constraints m. While their dependence on other parameters suggests no overall speedup over classical methodologies, some quantum SDO solvers provide speedups in the low-precision regime. We exploit this fact to our advantage, and present an iterative refinement scheme for the Hamiltonian Updates algorithm of Brandão et al. (*Quantum* 6, 625 (2022)) to exponentially improve the dependence of their algorithm on the precision ϵ, defined as the absolute gap between primal and dual solution. As a result, we obtain a classical algorithm to solve the semidefinite relaxation of Quadratic Unconstrained Binary Optimization problems (QUBOs) in matrix multiplication time. Provided access to a quantum read/classical write random access memory (QRAM), a quantum implementation of our algorithm exhibits $\mathcal{O}\left(ns + n^{1.5} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$ running time, where C is the cost matrix, $\|C\|_F$ is its Frobenius norm, and s is its sparsity parameter (maximum number of nonzero elements per row).

1 Introduction

We consider optimization problems of the form:

$$\begin{aligned} \max \quad & x^\top C x \\ \text{subject to} \quad & x \in \{-1, 1\}^n, \end{aligned} \tag{1}$$

where $C \in \mathcal{S}^n$ is the problem data and $\mathcal{S}^n$ is the space of symmetric matrices in $\mathbb{R}^{n \times n}$. Solving (1) can be viewed as computing the $\infty \to 1$ norm of the coefficient matrix C. This particular norm is intrinsically related to the *cut norm* of a matrix, which plays a crucial role in developing efficient approximation algorithms for dense graph and matrix problems [2, 22], with perhaps the most well-known application being the task of finding the largest cut in a graph (MaxCut). These problems also play an important role in quantum information sciences; the Ising model belongs to this class of problems [55], and quantum algorithms such as the Quantum Approximate Optimization Algorithm (QAOA) [19] and quantum annealing [20] can address its solution.

Computing the cut norm corresponds to replacing $x \in \{-1, 1\}^n$ with $z \in \{0, 1\}^n$ in (1), giving rise to *quadratic unconstrained binary optimization* (QUBO) problems. A standard QUBO is of the form

$$\begin{aligned} \max \quad & z^\top C z \\ \text{subject to} \quad & z \in \{0, 1\}^n. \end{aligned} \tag{2}$$

*Corresponding Author: bra216@lehigh.edu

†Department of Industrial and Systems Engineering, Quantum Computing and Optimization Lab, Lehigh University

‡Department of Industrial and Systems Engineering, University of Southern California

Provided that we allow for linear terms (in both formulations), it is well known that solutions to (1) can be used to compute a solution to (2) which differs only by a constant factor, and vice-versa, due to the equivalence $z = \frac{x+e}{2}$ if $z \in \{0,1\}^n$ and $x \in \{-1,1\}^n$, where $e \in \mathbb{R}^n$ is the all ones vector of dimension n.

Although (1) and (2) cover many applications of interest, they are intrinsically difficult to solve; computing optimal solutions to either (1) or (2) is NP-Hard in general. Following the seminal work of Lovász [44] and the theoretical and practical development of Interior Point Methods (IPMs) for solving semidefinite optimization (SDO) problems [47, 50, 51, 52, 53, 59, 60], a prevailing approach has been to obtain approximate solutions to (1) and (2) by relaxing integrality and lifting the problem from a vector space of dimension n, to the space of $n \times n$ symmetric matrices. The quadratic form $x^\top Cx$ can be equivalently expressed by $\operatorname{tr}(Cxx^\top)$, where $\operatorname{tr}(U)$ denotes the sum of the diagonal elements (or, trace) of a matrix $U \in \mathbb{R}^{n\times n}$. To deal with the bilinear term $xx^\top$, we introduce a matrix variable $X \in \mathbb{R}^{n\times n}$, and require that X satisfies the following:

$$\operatorname{diag}(X) = e, \quad X \succeq 0, \quad \operatorname{rank}(X) = 1,$$

where the notation $U \succeq V$ means that the matrix $U - V$ is a symmetric positive semidefinite matrix. Under these requirements, X is guaranteed to be of the form $X = xx^\top$ for $x \in \{-1,1\}^n$. The rank constraint, however, is not convex, and thus dropping it yields the following (convex) SDO relaxation of (1):

$$\begin{aligned} \max \quad & \operatorname{tr}(CX) \\ \text{subject to} \quad & \operatorname{diag}(X) = e, \quad X \succeq 0. \end{aligned} \tag{3}$$

Although the optimal solution X^* to (3) is no longer guaranteed to satisfy $X^* = x^*{x^*}^\top$ and may not be integral in general, the approximation of x^* provided by X^* is of sufficient quality to justify its use. In fact, SDO approximations cover some of the most celebrated results in optimization, such as the 0.878-approximation guarantee of Goemans and Williamson for MaxCut [29] and the Lovász-ϑ number [44].

1.1 Literature Review

More generally, a (primal) SDO problem involving $n \times n$ matrices and m constraints is of the form

$$\begin{aligned} \sup_X \quad & \operatorname{tr}(CX) \\ \text{subject to} \quad & \operatorname{tr}(A_i X) = b_i \quad \text{for } i \in [m], \\ & X \succeq 0, \end{aligned}$$

where $[m] = \{1, \ldots, m\}$ and $A_1, \ldots, A_m, C \in \mathcal{S}^n$, and $b \in \mathbb{R}^m$ are the (given) problem data. The *dual* SDO problem associated with the primal is given by

$$\begin{aligned} \inf_{(u,S)} \quad & b^\top u \\ \text{subject to} \quad & S = \sum_{i=1}^m u_i A_i - C \succeq 0. \end{aligned}$$

where S is the dual slack matrix.[1] The classical literature on algorithms for solving SDO problems is rich and can be categorized into two classes; algorithms that depend poly-logarithimically on the inverse precision to which we solve the problem and the size of the minimally inscribed ellipsoid, and algorithms that depend polynomially on these quantities but exhibit an advantage with respect to n and m. For instances with $m \leq \sqrt{n}$, the cutting plane methods (CPMs) of [36, 43] are the best performing classical algorithms,[2] and can solve SDO problems in time

$$\mathcal{O}\left(m(mns + m^2 + n^\omega) \cdot \operatorname{polylog}\left(m, n, R, \frac{1}{\epsilon}\right)\right),$$

[1] While the dual variable is typically denoted by y rather than u, it is also customary in the literature to use y to denote a certain state preparation pair, and we do so later in this paper.

[2] We remark that the running time in [36] does however exhibit improved dependence with respect to poly-logarithmic factors compared to the running time of [43].

where $\omega \in [2, 2.38]$ is the matrix multiplication exponent, R is an upper bound on the trace of a primal optimal solution X (which can be exponentially large, see [56]), ϵ is the precision parameter, s denotes the maximum number of nonzeros per row of the input matrices and hence, $\mathcal{O}(mns)$ is the total number of nonzeros in the constraints of SDO problem. However, we typically have $m \in [\Omega(n), \mathcal{O}(n^2)]$, in which case the CPMs given in [36, 43] are outperformed by the IPM for SDO from Jiang et al. [35]. Their IPM exhibits a worst case running time of

$$\mathcal{O}\left(\sqrt{n}(mns + m^\omega + n^\omega) \cdot \text{polylog}\left(m, n, \frac{1}{\epsilon}\right)\right),$$

where the term $m^\omega + n^\omega$ represents the per-iteration cost of inverting the Hessian and matrices of the variables.

While quantum SDO solvers could also be categorized in a somewhat similar fashion, it is perhaps more natural to do so according to how they attempt to obtain quantum speedups. In this case we also have two classes; at a high level, all proposed quantum SDO solution methodologies quantize a classical algorithm by either using quantum linear system algorithms (QLSAs) [13, 15, 32], or a quantum mechanical interpretation of normalized positive semidefinite matrices. We now review these works in detail.

The former class is comprised of algorithms that quantize IPMs, giving rise to quantum IPMs (QIPMs). QIPMs attempt to speedup the bottleneck of the classical IPM by substituting the classical solution of the Newton linear system with the combined use of a QLSA and quantum state tomography (with some classical computation between iterates). Augustino et al. [7] present a convergent QIPM for SDO, avoiding the shortcomings prevalent in early works on QIPMs (see, e.g., [40]), by properly symmetrizing the Newton linear system, and utilizing an orthogonal subspace representation of the search directions. This representation guarantees that primal and dual feasibility are satisfied exactly by all the iterates generated by inexact solutions of the Newton linear system obtained via quantum subroutines. The worst case complexity of their algorithm is

$$\widetilde{\mathcal{O}}_{n,\kappa,\frac{1}{\epsilon}}\left(\sqrt{n}\left(\frac{n^3\kappa^2}{\epsilon} + n^4\right)\right),$$

where κ is an upper bound on the condition numbers of the intermediate Newton linear system coefficient matrices that arise over the course of the algorithm. Here, the notation $\widetilde{\mathcal{O}}_{a,b}(f(x))$ suppresses poly-logarithmic factors in $f(x)$, a and b that appear in the overall running time, i.e., $\widetilde{\mathcal{O}}_{a,b}(f(x)) \equiv \mathcal{O}(f(x) \cdot \text{polylog}(a, b, f(x)))$. While this QIPM achieves a speedup in n over the IPM from [35] when $m = \mathcal{O}(n^2)$, its dependence on κ and ϵ suggest no quantum advantage overall: the complexity of the classical IPM does not depend on κ and its dependence on ϵ^{-1} is logarithmic. As the authors in [7] note, dependence on the condition number bound κ is particularly problematic in the context of IPMs.

The second class of quantum SDO solvers are those that quantize algorithms based on matrix exponentials and Gibbs states. The most prominent example is the Matrix Multiplicative Weights Update (MMWU) Method of Arora and Kale [4], which can solve SDO problems in time

$$\widetilde{\mathcal{O}}_{n,R,\frac{1}{\epsilon}}\left(nms\left(\frac{Rr}{\epsilon}\right)^4 + ns\left(\frac{Rr}{\epsilon}\right)^7\right),$$

where r is a *known* ℓ_1-norm upper bound[3] on a dual optimal solution u. Unlike IPMs, the MMWU framework does not involve the solution of linear systems; rather, these algorithms alternate between candidate solutions to the primal and dual SDO problems. IPMs and MMWUs also employ different definitions of optimality; for IPMs, ϵ-optimality implies that the primal and dual feasible solutions exhibit a *normalized* duality gap bounded by ϵ, i.e.:

$$\frac{\text{tr}\,(XS)}{n} \leq \epsilon,$$

whereas an ϵ-optimal solution obtained using an MMWU approximates the optimal objective value to additive error ϵ (via binary search). Finally, we point out a distinction between these algorithms with respect to

[3] It is also assumed that $R, r \geq 1$.

output. While primal-dual IPMs return the primal-dual optimal solution (X, u, S), MMWUs report u, but may avoid explicitly reporting X and S to maintain the speedups they offer with respect to n. Reporting X under the MMWU framework necessitates the computation of matrix exponentials, which may impose a considerable overhead because it generally resorts to matrix multiplication.

The MMWU framework has been specialized to solve SDO problems of the form in (3) (see, e.g., [5]), and the current state of the art is attributed to Lee and Padmanabhan [42], who give an algorithm that can solve (3) to additive error $\|C\|_{\ell_1}\epsilon$ with overall complexity

$$\widetilde{\mathcal{O}}_{n,\frac{1}{\epsilon}}\left(ns\epsilon^{-3.5}\right),$$

where $\|C\|_{\ell_1} = \sum_{i,j}|C_{ij}|$. It is important to note however, that to achieve the stated complexity their methodology does not explicitly report[4] the solution X and the authors assume $\sum_{i,j}|C_{ij}| = n$. To achieve the same error scaling as the algorithms we present in this work, the algorithm in [42] would have overall cost $\widetilde{\mathcal{O}}_{n,\frac{1}{\epsilon}}\left(\|C\|_{\ell_1}^{3.5}ns\epsilon^{-3.5}\right)$, see Section 5.3.

Brandão and Svore [12] were the first to quantize the MMWU framework, utilizing a clever interpretation of the primal variables: *Gibbs states*, which can be efficiently prepared on a quantum computer, naturally correspond to trace-normalized positive definite matrices. The running time of these MMWU-based algorithms was subsequently improved [10, 31, 64, 65], and the current state of the art running time of the quantum MMWU (QMMWU) algorithm for SDO problems is:

$$\widetilde{\mathcal{O}}_{n,s,R,\frac{1}{\epsilon}}\left(\left(\sqrt{m}+\sqrt{n}\frac{Rr}{\epsilon}\right)s\left(\frac{Rr}{\epsilon}\right)^4\right).$$

Similar to the complexity of QIPMs, QMMWU algorithms are faster with respect to m and n when compared to their classical counterparts, but these algorithms still exhibit a non-polynomial running time, due to their polynomial dependence on the scale invariant parameter $\frac{Rr}{\epsilon}$, whereas the natural input size depends on the logarithm of this quantity.

Seeking to improve the performance of quantum SDO solvers, Brandão et al. [11] present an algorithm, which they call *Hamiltonian Updates* (HU), for solving the SDO approximation (3) of (1). The HU method is a primal-only algorithm closely related to the QMMWU framework, in that it leverages a Gibbs state representation of the primal variable and progression towards the optimal solution is made via matrix-exponentiated gradient updates. Specifically, the authors in [11] are interested in solving an SDO feasibility problem that arises upon renormalizing and relaxing (3):

$$\begin{aligned} \text{find} \quad & X \\ \text{subject to} \quad & \operatorname{tr}\left(\frac{C}{\|C\|}X\right) \geq \gamma - \epsilon \\ & \sum_{i\in[n]}\left|\langle i|X|i\rangle - \frac{1}{n}\right| \leq \epsilon \\ & \operatorname{tr}(X) = 1, \quad X \succeq 0. \end{aligned} \tag{4}$$

Here, γ is an upper bound on the absolute value of the optimal objective value of (3) when the cost matrix C is normalized, obtained via binary search over $[-1, 1]$, and $|i\rangle$ for $i \in \{1, \ldots, n\}$ are the computational basis states. Since any $\log(n)$-qubit Gibbs state is an element of the set $\{X \in \mathbb{R}^{n\times n} : \operatorname{tr}(X) = 1, X \succeq 0\}$ by definition, solutions to (4) can be naturally be expressed as a Gibbs state

$$\rho = \frac{\exp(-H)}{\operatorname{tr}(\exp(-H))},$$

where H is the *Hamiltonian* associated with ρ. The key observation in [11] is that upon using the Gibbs state change of variables in (4), one can model the n constraints on the diagonal elements as single constraint

[4] Alternatively, they report a "gradient" $G \in \mathcal{S}^n$ such that $X = W\exp(G)W$ for a diagonal matrix W.

which requires that the distribution on the diagonal elements of a feasible solution ρ to (4) be at most ϵ in total variation distance to the uniform distribution. In other words, the task of solving (4) reduces to finding a $\log(n)$-qubit mixed quantum state that upon measurement in the computational basis is approximately indistinguishable from the maximally-mixed state, and whose trace inner product with the normalized cost matrix $C\|C\|^{-1}$ is at least $\gamma - \epsilon$.

Using a quantum computer, the HU method of [11] solves (3) to additive error $\mathcal{O}(n\|C\|\epsilon)$ in time

$$\widetilde{\mathcal{O}}_{n,\frac{1}{\epsilon}}\left(n^{1.5}\sqrt{s}^{1+o(1)}\epsilon^{-28+o(1)}\exp\left(1.6\sqrt{\log(\epsilon^{-1})}\right)\right).$$

The authors in [11] also provide an analysis of essentially the same algorithm when using a classical computer, and show that the classical algorithm has a complexity of

$$\widetilde{\mathcal{O}}_n\left(\min\{n^2 s, n^\omega\}\epsilon^{-12}\right).$$

The quantum algorithm yields a speedup in n over classical algorithms, for a specific class of SDO problems. However, as we have already seen with QIPMs and QMMWU algorithms, its dependence on other parameters (in this case the inverse precision) is prohibitive unless a very low precision solution is acceptable. This raises the question as to whether the poor scaling in the inverse precision can be mitigated without incurring additional cost in n and s. We answer this question in the affirmative using iterative refinement techniques.

Iterative Refinement (IR) is a methodology for computing high-precision solutions to linear system of equations [30], as well as linear [26, 27, 28] and mixed integer optimization problems [3, 18]. We summarize the methodology at a high level as follows, and present a detailed discussion for the case of convex feasibility problems later in the paper. Given an initial solution $x^{(0)} \in \mathbb{R}^d$, at each iteration k IR produces a refined solution $x^{(k+1)} \leftarrow x^{(k)} + u^{(k)}$, where $u^{(k)}$ acts as a correction of the error $r^{(k)}$ associated with $x^{(k)}$, and is determined by solving a *refining problem* induced by the current solution. These operations can all be carried out using the same level of accuracy, called the *fixed precision* approach. Alternatively, one may increase the accuracy with which the residuals $r^{(k)}$ are computed as compared to $u^{(k)}$, and this approach is called a *mixed precision* approach [30, 66]. In this paper, we utilize the fixed precision approach.

1.2 Contributions

In this paper we develop an IR scheme for SDO approximations of QUBO problems that uses the HU algorithm of [11] as a subroutine. We show that proceeding in this way allows one to exponentially improve the dependence on the inverse precision for both the quantum and classical algorithms.

With the proposed IR scheme, the classical algorithm solves the SDO problem (3) up to absolute error $\mathcal{O}(\epsilon)$ with worst-case complexity

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\}\cdot\operatorname{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right).$$

This is a significant speedup compared to general-purpose SDO solvers, such as IPMs. This algorithm can be quantized following a similar strategy to [11]. When provided access to quantum random access memory (QRAM), the quantum algorithm takes

$$\mathcal{O}\left(n^{1.5}\cdot\operatorname{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$$

accesses to the QRAM and additional quantum gates (this is the standard way of describing complexity in the QRAM model of computation), plus $\mathcal{O}(ns)$ classical arithmetic operations — note that simply reading the cost matrix C takes $\mathcal{O}(ns)$ time.

Summarizing, the combination of HU with IR described in this paper provides exponential speedups over the methodology proposed in [11] with respect to the precision parameter ϵ. To the best of our knowledge, our classical and quantum algorithms are the fastest known algorithms in their respective model of computation for this class of problems, and our quantum algorithm provides a genuine asymptotic speedup over

known classical solution methodologies, provided that we have access to QRAM. In the sparse-access input model (without QRAM), the algorithm takes $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(n^{1.5}s^{0.5+o(1)}\right)$ accesses to an oracle describing the coefficient matrix C and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(n^{2.5}s^{0.5+o(1)}\right)$ additional gates, therefore yielding no quantum speedup (the quantum gate complexity is asymptotically larger than the classical complexity).

The remainder of this paper is organized in the following manner. Section 2 introduces notation, as well as the relevant input models and quantum subroutines. In Section 3 we introduce the Hamiltonian Updates (HU) algorithm from [11], and our Iterative Refinement scheme for SDO approximations of QUBOs is presented in Section 4. The running time analysis is performed in Section 5, and Section 6 concludes the manuscript.

2 Preliminaries

We write $[n]$ to represent the set of elements $\{1,\ldots,n\}$. We denote the i-th element of a vector $x\in\mathbb{R}^n$ by x_i for $i\in[n]$, and the ij-th element of a matrix $A\in\mathbb{R}^{m\times n}$ by A_{ij} for $i\in[m]$ and $j\in[n]$. To refer to the i-th row of a matrix A, we write $A_{i,\cdot}$ and write $A_{\cdot,j}$ when referring to its j-th column. We distinguish the quantity a to the k-th power and the value of a at iterate k using round brackets, writing a^k and $a^{(k)}$ to denote these quantities, respectively.

The smallest and largest singular values of a matrix A are denoted $\sigma_{\min}(A),\sigma_{\max}(A)$, and the smallest and largest eigenvalues are denoted $\lambda_{\min}(A),\lambda_{\max}(A)$. We let $\mathcal{S}^n_+$ and $\mathcal{S}^n_{++}$ represent the cones of symmetric positive semidefinite, and symmetric positive definite matrices, respectively. For $A,B\in\mathcal{S}^n$, we write $A\succeq B$ ($A\succ B$) to indicate that the matrix $A-B$ is symmetric positive semidefinite (symmetric positive definite), i.e., $A-B\in\mathcal{S}^n_+$ ($A-B\in\mathcal{S}^n_{++}$). The matrix exponential $\exp(A)$, which is defined by the power series

$$\exp(A) := I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \cdots,$$

maps symmetric matrices to the space of symmetric positive definite matrices. Given the spectral decomposition $A=V\Lambda V^\top$, then $\exp(A)=V\exp(\Lambda)V^\top$, where $\exp(\Lambda)=\operatorname{diag}(\exp(\Lambda_{11}),\exp(\Lambda_{22})\ldots,\exp(\Lambda_{nn}))$.

We let $A\circ B$ denote the Hadamard (or element-wise) product of two matrices, and $A\otimes B$ denotes their tensor product. Later in this work, we make use of the following facts regarding Hadamard products.

Lemma 1 (Lemma 5.1.4 in [34])**.** *Let E, F and G be $m\times n$ matrices. Then, the i-th diagonal entry of the matrix $(E\circ F)G^\top$ coincides with the i-th diagonal entry of the matrix $(E\circ G)F^\top$. That is,*

$$[(E\circ F)G^\top)]_{ii} = [(E\circ G)F^\top)]_{ii} \quad \forall i\in[m].$$

Lemma 2 (Theorem 5.3.4 in [34])**.** *Let A and B be $n\times n$ Hermitian matrices. If $A\in\mathcal{S}^n_+$, then any eigenvalue $\lambda(A\circ B)$ of $A\circ B$ satisfies*

$$\lambda_{\min}(A)\cdot\lambda_{\min}(B)\le\min_{i\in[n]}A_{ii}\cdot\lambda_{\min}(B)\le\lambda(A\circ B)\le\max_{i\in[n]}A_{ii}\cdot\lambda_{\min}(B)\le\lambda_{\max}(A)\cdot\lambda_{\max}(B).$$

We write e to refer to the vector of all ones in $\mathbb{R}^n$, and use the notation e_i to refer to the i-th unit vector in the standard orthonormal basis $\{e_1,\ldots,e_n\}$ for $\mathbb{R}^n$. Analogously, the computational basis states are denoted by $|i\rangle$ for $i\in[n]$. Hence, for $x\in\mathbb{R}^n$, we denote its amplitude encoding by $|x\rangle$, defined as

$$|x\rangle = \frac{1}{\|x\|}\sum_{i\in[n]}x_i\,|i\rangle\,.$$

Observe that $|x\rangle$ is a $\log(n)$-qubit state; for simplicity, we assume that the dimensions of all spaces are powers of 2. All logarithms are base 2.

Where appropriate, our analysis makes use of the *Schatten p-norm*, defined for a bounded linear operator A as

$$\|A\|_p := \left[\operatorname{tr}\left(|A|^p\right)\right]^{\frac{1}{p}},$$

where $|A| = (A^\dagger A)^{\frac{1}{2}}$ with $A^\dagger$ denoting the conjugate transpose of A. Notice that the trace and operator norms $\|\cdot\|_{\mathrm{tr}}$ and $\|\cdot\|$ are the Schatten-1 and Schatten-∞ norms, respectively, and the Frobenius norm $\|\cdot\|_F$ corresponds to the Schatten-2 norm. Positive semidefinite matrices $A \in \mathcal{S}_+^n$ admit the useful identity $\|A\|_{\mathrm{tr}} = \mathrm{tr}(A) = \sum_{i\in[n]} \lambda_i(A)$, where $\lambda_i(A)$ is the i-th eigenvalue of A. The equivalence is due to the fact that the trace norm $\|A\|_{\mathrm{tr}} = \mathrm{tr}\left(\sqrt{A^\dagger A}\right)$ is defined as the sum of the singular values of A, and the singular values of A are equivalent to the eigenvalues of A whenever $A \in \mathcal{S}_+^n$.

For a scalar $x \in \mathbb{R}$ define the *sign function* $\mathrm{sign}(x)$ as

$$\mathrm{sign}(x) := \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0. \end{cases}$$

When $x \in \mathbb{R}^n$, $\mathrm{sign}(x) = (\mathrm{sign}(x_1), \ldots, \mathrm{sign}(x_n))^\top$.

For any positive integer q, and binary strings $j, k \in \{0,1\}^q$, we denote by $j \oplus k$ the bitwise modulo 2 addition of q-digit strings, defined as

$$j \oplus k := h$$

where $h \in \{0,1\}^q$ is the bitstring whose elements h_p are defined for $p \in [q]$ as

$$h_p := \begin{cases} 0 & \text{if } j_p = k_p, \\ 1 & \text{otherwise.} \end{cases}$$

"Big-O" notation

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

$$f(x) = \mathcal{O}(g(x)) \iff \exists \ell \in \mathbb{R}, c \in \mathbb{R}_+, \text{ such that } f(x) \leq cg(x) \quad \forall x > \ell.$$

We write $f(x) = \Omega(g(x)) \iff g(x) = \mathcal{O}(f(x))$. We also define $\widetilde{\mathcal{O}}(f(x)) = \mathcal{O}(f(x) \cdot \mathrm{polylog}(f(x)))$ and when the function depends poly-logarithmically on other variables we write

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

2.1 Input models and subroutines

For our quantum algorithm, we provide analyses for two distinct models of input. One model considers a *quantum-read/classical-write* RAM (QRAM), and the other is the *sparse-access model*, which we use to bound the running time without access to QRAM.

2.1.1 Sparse-access model

In the *sparse-access model*, the input matrix C is assumed to be s-row sparse for some known bound $s \in [n]$. In other words, C has at most s nonzero entries per row. The sparse-access model is closely related to the classical notion, in that we assume access to an oracle O_{sparse}, which upon being queried with input (i, j) returns the index of the j-th nonzero entry of the i-th row of C by calculating the index function:

$$\mathrm{index} : [n] \times [s] \to [n].$$

That is, for $i \in [n]$ and $j \in [s]$, O_{sparse} computes the position in place:

$$O_{\mathrm{sparse}} |i, j\rangle = |i, \mathrm{index}(i, j)\rangle .$$

We also assume access to an oracle that returns a bitstring representation of the individual entries of the normalized cost matrix $C\|C\|_F^{-1}$ for every $i, j \in [n]$:

$$O_C |i, j, z\rangle = \left|i, j, z \oplus (C_{ij}\|C\|_F^{-1})\right\rangle .$$

2.1.2 Quantum random access memory

We consider a *quantum-read/classical-write* RAM (QRAM), which enables us to store classical data that our quantum algorithms can make oracle calls to. This type of storage is the direct quantum analog of classical RAM: it enables a quantum algorithm to access classical data in superposition. Accessing a QRAM of size n takes $\mathcal{O}(n)$ gates [6, 25], but these gates can be arranged in parallel so that the circuit depth remains $\mathcal{O}(\text{polylog}(n))$. Therefore we make the assumption (standard in the literature on quantum algorithms) that the cost of accessing a QRAM of size n is $\mathcal{O}(\text{polylog}(n))$.

The next result from Chakraborty et al. [13], is adapted from an earlier result of Kerenidis and Prakash [39] and summarizes the aspects of the data structure we utilize.

Theorem 1 (Theorem 1 in [13])**.** *(Implementing quantum operators using an efficient data structure) Let $A \in \mathbb{R}^{m\times n}$ be a matrix. If w is the number of non-zero entries of A, then there exists a data structure of size $\mathcal{O}\left(w \log^2(mn)\right)$ that, given the entries (i, j, A_{ij}) in an arbitrary order, stores them such that time taken to store each entry of A is $\mathcal{O}(\log(mn))$. Once this data structure has been initiated with all non-zero entries of A, there exists a quantum algorithm that can perform the following maps with ξ-precision in time $\mathcal{O}\left(\text{polylog}\left(\frac{mn}{\xi}\right)\right)$:*

$$\widetilde{U} : |i\rangle |0\rangle \mapsto |i\rangle \frac{1}{\|A_{i,\cdot}\|} \sum_{j=1}^{n} A_{ij} |j\rangle = |i, A_{i,\cdot}\rangle ,$$

$$\widetilde{V} : |0\rangle |j\rangle \mapsto \frac{1}{\|A\|_F} \sum_{i=1}^{m} \|A_{i,\cdot}\| |i\rangle |j\rangle = \left|\widetilde{A}, j\right\rangle ,$$

where $|A_{i,\cdot}\rangle$ is the normalized quantum state corresponding to the i-th row of A and $\left|\widetilde{A}\right\rangle$ is a normalized quantum state such that $\langle i|\widetilde{A}\rangle = \|A_{i,\cdot}\|$, i.e., the norm of the i-th row of A.

2.1.3 Working with block-encoded matrices

We now give a formal definition of a block-encoding from [13].

Definition 1 (Block-encoding)**.** *Let $A \in \mathbb{C}^{2^w \times 2^w}$ be a w-qubit operator. Then, a $(w + a)$-qubit unitary U is an (α, a, ξ)-block-encoding of A if $U = \begin{pmatrix} \widetilde{A} & \cdot \\ \cdot & \cdot \end{pmatrix}$, with the property that*

$$\|\alpha\widetilde{A} - A\| \leq \xi.$$

It was shown by Kerenidis and Prakash [39] and Chakraborty et al. [13] how to efficiently implement block-encodings of matrices that are stored in a QRAM data structure, which is formalized in the next result.

Lemma 3 (Lemma 3.3.7 in [23])**.** *Let $A \in \mathbb{C}^{2^w \times 2^w}$ and $\xi > 0$.*

(i) Fix $q \in [0, 2]$ and define $\mu_q(A) = \sqrt{n_q(A) n_{(2-q)}(A^\top)}$ where $n_q(A) = \max_i \|A_{i,\cdot}\|_q^q$ is the q-th power of the maximum q-norm of the rows of A. Defining $A^{\{q\}}$ to be the matrix with elements $A_{ij}^{\{q\}} = \sqrt{A_{ij}^q}$, if $A^{\{q\}}$ and $(A^{\{2-q\}})^\dagger$ are both stored in QRAM data structures, then there exist unitaries U_R and U_L that can be implemented in time $\mathcal{O}(\text{poly}(w \log \frac{1}{\xi}))$ and such that $U_R^\dagger U_L$ is a $(\mu_q(A), w + 2, \xi)$-block-encoding of A.

(ii) If A is stored in a QRAM data structure, then there exist unitaries U_R and U_L that can be implemented in time $\mathcal{O}(\text{poly}(w \log \frac{1}{\xi}))$ and such that $U_R^\dagger U_L$ is an $(\|A\|_F, w + 2, \xi)$-block-encoding of A.

Linear combinations of block-encodings can also be constructed at cost that is merely logarithmic in the dimension.

Definition 2 (Definition 3.3.8 in [23]). *(State preparation pair) Let $y \in \mathbb{C}^m$ and $\|y\|_1 \leq \beta$. The pair of unitaries (P_L, P_R) is called a (β, p, ξ)-state-preparation-pair if $P_L |0\rangle^{\otimes p} = \sum_{j=0}^{2^p-1} c_j |j\rangle$ and $P_R |0\rangle^{\otimes p} = \sum_{j=1}^{2^p-1} d_j |j\rangle$ such that $\sum_{j=0}^{m-1} |\beta(c_j^* d_j) - y_j| \leq \xi$ and for all $j \in m, \ldots, 2^p - 1$ we have $c_j^* d_j = 0$.*

Proposition 1 (Lemma 52 in [24]). *(Linear combination of block-encoded matrices, with weights given by a state preparation pair) Let $A = \sum_{j=0}^{m-1} y_j A_j$ be a w-qubit operator, where A_j are matrices. Suppose P_L, P_R is a (β, p, ξ_1)-state-preparation pair for y, $W = \sum_{j=0}^{m-1} |j\rangle\langle j| \otimes U_j + ((I - \sum_{j=0}^{m-1} |j\rangle\langle j|) \otimes I_a \otimes I_s)$ is an $(w + a + p)$-qubit unitary with the property that U_j is an (α, a, ξ_2)-block-encoding of A_j. Then we can implement a $(\alpha\beta, a + p, \alpha\xi_1 + \alpha\beta\xi_2)$-block-encoding of A with a single use of W, P_R and $P_L^\dagger$.*

It turns out that the sparse-access model reduces to the quantum operator model upon choosing $\alpha = s$ (if row and column sparsity are the same). The next result from [24] describes how to implement block-encodings using the sparse-access input model, and the associated costs.

Lemma 4 (Lemma 48 in [24]). *Let $A \in \mathbb{C}^{2^w \times 2^w}$ be a matrix that is s_r-row-sparse and s_c-column-sparse, and each element of A has absolute value at most 1. Suppose that we have access to the following sparse-access oracles acting on two $(w + 1)$ qubit registers:*

$$O_r : |i\rangle |k\rangle \mapsto |i\rangle |r_{ik}\rangle \quad \forall i \in [2^w] - 1, k \in [s_r], \text{ and}$$
$$O_c : |\ell\rangle |j\rangle \mapsto |c_{\ell j}\rangle |j\rangle \quad \forall \ell \in [s_c], j \in [2^w] - 1, \text{ where}$$

r_{ij} is the index for the j-th non-zero entry of the i-th row of A, or if there are less than i non-zero entries, then it is $j + 2^w$, and similarly c_{ij} is the index for the i-th non-zero entry of the j-th column of A, or if there are less than j non-zero entries, then it is $i + 2^w$. Additionally, assume that we have access to an oracle O_A that returns the entries of A in a binary description:

$$O_A : |i\rangle |j\rangle |0\rangle^{\otimes p} \mapsto |i\rangle |j\rangle |a_{ij}\rangle, \quad \forall i, j \in [2^w] - 1,$$

where a_{ij} is a p-bit binary description of the ij-matrix element of A. Then, we can implement a $(\sqrt{s_r s_c}, w + 3, \xi)$-block-encoding of A with a single use of O_r, O_c and two uses of O_A, and additionally using $\mathcal{O}\left(w + \log^{2.5}\left(\frac{s_r s_c}{\xi}\right)\right)$ one and two qubit gates while using $\mathcal{O}\left(p + \log^{2.5}\left(\frac{s_r s_c}{\xi}\right)\right)$ ancilla qubits.

The block-encoding framework will be useful in speeding up the overall running time found in [11], as it allows us to perform matrix computations and Hamiltonian simulation efficiently.

Theorem 2 (Corollary 3.4.7 in [23]). *(Optimal block-Hamiltonian simulation) Suppose that U is an $(\alpha, a, \xi/|2t|)$-block-encoding of the Hamiltonian H. Then, we can implement a ξ-precise Hamiltonian simulation unitary V which is an $(1, a + 2, \xi)$-block-encoding of e^{itH}, with $\mathcal{O}\left(|\alpha t| + \frac{\log(1/\xi)}{\log\log(1/\xi)}\right)$ uses of controlled-U or its inverse and with $\mathcal{O}\left(a|\alpha t| + a\frac{\log(1/\xi)}{\log\log(1/\xi)}\right)$ two-qubit gates.*

Additionally, one can easily take the product of block-encodings.

Proposition 2 (Lemma 4 in [13]). *(Product of block-encoded matrices) If U_A is an (α_1, a_1, ξ_A)-block-encoding of an s-qubit operator A, and U_B is an (α_2, a_2, ξ_B)-block-encoding of an s-qubit operator B, then $(I_{a_2} \otimes U_A)(I_{a_1} \otimes U_B)$ is an $(\alpha_1\alpha_2, a_1 + a_2, \alpha_1\xi_B + \alpha_2\xi_A)$-block-encoding of AB.*

Relevant to our work in the quantum operator input model is the idea of block-encoding the *Hadamard*, or element-wise product of two matrices. We will demonstrate how one can carry out the Hadamard product of block-encodings of matrices A and B as a reduction of the Kronecker product of block-encodings, which is straightforward to construct given block encodings of A and B.

Proposition 3. *(Kronecker product of block-encoded matrices) Suppose that U_A is an (α_1, a_1, ξ_A)-block-encoding of $A \in \mathbb{R}^{n \times n}$, and U_B is an (α_2, a_2, ξ_B)-block-encoding of $B \in \mathbb{R}^{n \times n}$. Then, taking the tensor product of U_A and U_B, we obtain a $(\alpha_1\alpha_2, a_1 + a_2, \xi_A + \xi_B)$-block-encoding of $A \otimes B$.*

We do not give a formal proof here as the result directly follows from the definition of a block-encoding; to obtain the tensor product of two block-encoded matrices, it suffices to take the tensor product of their block-encodings while keeping the ancilla qubits separate.

Proposition 4. *(Hadamard product of block-encoded matrices) Suppose that U_A is an (α_1, a_1, ξ_A)-block-encoding of $A \in \mathbb{R}^{n\times n}$, and U_B is a (α_2, a_2, ξ_B)-block-encoding $B \in \mathbb{R}^{n\times n}$. Then, using U_A and U_B, we can implement an $(\alpha_1\alpha_2, a_1 + a_2 + 8\log(n) + 12, 5(\xi_A + \xi_B))$-block-encoding of $A \circ B$ using one application of U_A and U_B, and $\widetilde{\mathcal{O}}_n(1)$ additional gates.*

Proof. First, note that

$$A \circ B = (A \otimes B)[\iota_A, \iota_B],$$

where $\iota_A = \iota_B = \{1, n+2, 2n+3, \ldots, n^2\}$ are index sets of cardinality n (see, e.g., Lemma 5.1.1 in [34]). Our goal is to use the index sets ι_A and ι_B along with a block encoding of $A \otimes B$ to construct a unitary which block-encodes $\mathcal{M} \in \mathbb{R}^{n^2\times n^2}$, a matrix which contains the elements of $A \circ B$ in its upper left-most $n \times n$ block, while all other entries are 0:

$$\mathcal{M}_{ij} = \begin{cases} A_{ij} \cdot B_{ij} & \text{for } i, j = 1, \ldots, n, \\ 0 & \text{otherwise,} \end{cases}$$

i.e.,

$$\mathcal{M} = \begin{pmatrix} A \circ B & \mathbf{0}^{n\times(n^2-n)} \\ \mathbf{0}^{(n^2-n)\times n} & \mathbf{0}^{(n^2-n)\times(n^2-n)} \end{pmatrix}.$$

We will first show how one can use ι_A and ι_B to construct sparse matrices that map $A \otimes B$ to $\mathcal{M}$, and then subsequently analyze the cost of constructing the corresponding unitary block-encoding.

Consider the matrix $Z \in \mathbb{R}^{n^2\times n^2}$, whose elements are defined as

$$Z_{ij} = \begin{cases} 1 & \text{if } i = j = (k-1)n + k, \quad k = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

Multiplying $A \otimes B$ on the left by Z sets the rows of $A \otimes B$ which do not contain elements of $A \circ B$ to zero, and subsequently multiplying $Z(A \otimes B)$ on the right by Z will set the columns of $Z(A \otimes B)$ which do not appear in $A \circ B$ to zero. As a result, a block-encoding of $Z(A \otimes B)Z$ corresponds to block-encoding $A \otimes B$, and setting all terms not appearing in $A \circ B$ to zero:

$$[Z(A \otimes B)Z]_{ij} = \begin{cases} [A \otimes B]_{ij} & \text{if } i = (k-1)n + k \text{ and } j = (\ell-1)n + \ell \quad k, \ell = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

Next, let $G \in \mathbb{R}^{n^2\times n^2}$ be a matrix whose elements are defined as follows:

$$G_{ij} = \begin{cases} 1 & \text{if } i \in [n^2] \text{ and } i = j = (k-1)n + k, \quad k = 1, \ldots, n, \\ 1 & \text{if } i \in [n^2] \setminus \{1, n+2, 2n+3, \ldots, n^2\} \text{ and } j = (i-1)n + i, \\ 0 & \text{otherwise.} \end{cases}$$

We will now establish that $GZ(A \otimes B)Z)G^\top$ is precisely the matrix we seek to block-encode, by demonstrating that $G(Z(A \otimes B)Z)G^\top = \mathcal{M}$. First, observe that G is a (partial) permutation matrix: multiplying $Z(A \otimes B)Z$ on the left by G performs the necessary row-exchanges, as the elements of $G(Z(A \otimes B)Z)$ are given by

$$[G\,(Z(A \otimes B)Z)]_{ik} = \begin{cases} A_{ij} \cdot B_{ij} & \text{for } k = (j-1)n + j, \quad i, j = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

On the other hand, multiplying $Z(A \otimes B)Z)$ on the right by $G^\top$ performs this transformation with respect to the columns such that

$$[(Z(A \otimes B)Z)\, G]_{kj} = \begin{cases} A_{ij} \cdot B_{ij} & \text{for } k = (i-1)n + i, \quad i, j = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, multiplying $G\,(Z(A \otimes B)Z)$ on the right by $G^\top$ conducts the column exchanges to move $A \circ B$ to the top left n-dimensional block of $Z(A \otimes B)Z$, i.e.,

$$[G\,(Z(A \otimes B)Z)\, G]_{ij} = \begin{cases} A_{ij} \cdot B_{ij} & \text{for } i, j = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $G(Z(A \otimes B)Z)G^\top = \mathcal{M}$ as desired.

We now analyze the cost associated with block-encoding $\mathcal{M}$. Under the stated hypothesis, we have access to an (α_1, a_1, ξ_A)-block-encoding U_A of A, and an (α_2, a_2, ξ_B)-block-encoding U_B of B, and thus applying Proposition 3 we can construct an $(\alpha_1\alpha_2, a_1 + a_2, \xi_A + \xi_B)$-block-encoding $U_{A\otimes B}$ of $A \otimes B$ using one application of U_A and of U_B, and no additional gates.

Using the description of Z, we can construct the sparse-access oracles O_r and O_c as defined in Lemma 4 (which act on two $(2\log n + 1)$ qubit registers). Additionally, from the definition of Z, we can construct an oracle O_Z, which returns the entries of Z in a binary description:

$$O_Z : |i\rangle\,|j\rangle\,|0\rangle^{\otimes p} \mapsto |i\rangle\,|j\rangle\,|z_{ij}\rangle\,, \quad \forall i, j \in [2^{2\log n}] - 1,$$

where z_{ij} is a p-bit binary description of the ij-matrix element of Z. Note that the circuit for the position and value of the nonzero elements of Z using $\widetilde{\mathcal{O}}_n(1)$ gates because they admit an efficient description: their value is 1 and we have a compact description of their position. By construction the matrix Z is 1-row sparse and 1-column sparse, and hence an application of Lemma 4 with $s_r = s_c = 1$ asserts that one can construct a $(1, 2\log(n)+3, \xi_Z)$-block-encoding U_Z of Z. Given block-encodings U_Z and $U_{A\otimes B}$, we can apply Proposition 2 with

$$\xi_Z = \frac{\xi_A + \xi_B}{\alpha_1\alpha_2}, \quad \xi_{A\otimes B} = \xi_A + \xi_B,$$

yielding an $(\alpha_1\alpha_2, a_1 + a_2 + 2\log(n) + 3, 2(\xi_A + \xi_B))$-block-encoding of $Z(A \otimes B)$. Applying Proposition 2 once more with

$$\xi_Z = \frac{\xi_A + \xi_B}{\alpha_1\alpha_2}, \quad \xi_{Z(A\otimes B)} = 2(\xi_A + \xi_B),$$

we obtain an $(\alpha_1\alpha_2, a_1 + a_2 + 4\log(n) + 6, 3(\xi_A + \xi_B))$-block-encoding of $Z(A \otimes B)Z$.

Just as was the case with Z, we can use the description of G to construct the sparse-access oracles O_r and O_c as defined in Lemma 4 (which again, act on two $(2\log n + 1)$ qubit registers), as well as an oracle O_G using $\widetilde{\mathcal{O}}_n(1)$ gates, that returns the entries of G in a binary description:

$$O_G : |i\rangle\,|j\rangle\,|0\rangle^{\otimes p} \mapsto |i\rangle\,|j\rangle\,|g_{ij}\rangle\,, \quad \forall i, j \in [2^{2\log n}] - 1,$$

where g_{ij} is a p-bit binary description of G_{ij} (the ij-matrix element of G). Noting that G is 1-row sparse and 1-column sparse (and hence, so its transpose); applying Lemma 4 twice more allows us to construct a $(1, 2\log(n) + 3, \xi_G)$-block-encoding U_G of G, as well as a $(1, 2\log(n) + 3, \xi_G^\top)$-block-encoding $U_{G^\top}$ of the transpose $G^\top$. We can then use U_G and our $(\alpha_1\alpha_2, a_1+a_2+4\log(n)+6, 3(\xi_A+\xi_B))$-block-encoding $U_{Z(A\otimes B)Z}$ of $Z(A \otimes B)Z$ to construct an $(\alpha_1\alpha_2, a_1 + a_2 + 6\log(n) + 9, 4(\xi_A + \xi_B))$-block-encoding of $G(Z(A \otimes B)Z)$ by applying Proposition 2 with

$$\xi_G = \frac{\xi_A + \xi_B}{\alpha_1\alpha_2}, \quad \xi_{Z(A\otimes B)Z} = 3(\xi_A + \xi_B).$$

Applying Proposition 2 a final time, with

$$\xi_{G^\top} = \frac{\xi_A + \xi_B}{\alpha_1 \alpha_2}, \quad \xi_{G(Z(A\otimes B)Z)} = 4(\xi_A + \xi_B),$$

produces an $(\alpha_1\alpha_2, a_1 + a_2 + 8\log(n) + 12, 5(\xi_A + \xi_B))$-block-encoding $U_{\mathcal{M}}$ of $\mathcal{M} = G(Z(A \otimes B)Z)G^\top$.

The stated complexity result follows upon noting that the steps required to construct the unitary

$$U_{\mathcal{M}} = U_G U_Z U_{A\otimes B} U_Z U_{G^\top}$$

requires one application of $U_{A\otimes B}$ and one application of each of the other matrices. In turn, this amounts to 1 application of U_A and U_B each, plus the $\widetilde{\mathcal{O}}_n(1)$ gate cost of the remaining matrices U_G, U_Z and $U_{G^\top}$, and the proof is complete. □

We remark that a similar result to Proposition 4 was independently derived and discussed in the recent paper [14].

2.1.4 Gibbs Samplers and Trace Estimators

For clarity, we begin with a formal definition of a subnormalized density operators and their purifications.

Definition 3 (Definition 6.3.1 in [23]). *(Subnormalized density operators & Purification) A subnormalized density operator ρ is a positive semidefinite matrix of trace at most 1. A purification ϱ of a subnormalized density operator ρ is a 3-register pure state such that tracing out the third register and projecting on the subspace where the second register is $|0\rangle$ yields ρ.*

The frameworks introduced later in this paper require that we implement a Gibbs sampler and a trace estimator, which we define next.

Definition 4 (Definition 4.11 in [62]). *(Gibbs Sampler) A θ-precise Gibbs-sampler for the input matrix H, is a unitary that takes as input a data structure storing a Hamiltonian H and creates as output a purification of a θ-approximation (in trace distance) of the Gibbs state*

$$\rho = \frac{\exp(-H)}{\mathrm{tr}(\exp(-H))}.$$

We will use these approximate Gibbs states in order to check the diagonal entries of our solutions, as well as compute the trace inner products of matrices (or, expectation values), i.e., quantities of the form $\mathrm{tr}(A\rho)$.

Definition 5 (Definition 4.12 in [62]). *(Trace Estimator) A θ-precise trace estimator is a unitary that as input takes a state ρ and a matrix A. It outputs a sample from a random variable $x \in \mathbb{R}$ such that x is an estimator for $\mathrm{tr}(A\rho)$ that is at most $\theta/4$ biased.*

These implementations require polynomial approximations of the exponential function, which can be obtained using quantum singular value transformation techniques introduced in [23, 24].

Lemma 5 (Lemma 4.14 in [62]). *Let $\xi \in (0, 1/6]$ and $\beta \geq 1$. There exists a polynomial $P(x)$ such that*

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

Lemma 6 (Lemma 4.15 in [62]). *Let $\theta \in (0, 1/3]$, $\beta > 1$, and let d be the degree of the polynomial from Lemma 5 when we let $\xi = \frac{\theta}{128n}$. Let U be a $(\beta, a, \frac{\theta^2\beta}{1024^2 d^2 n^2})$-block-encoding of a Hermitian operator $H \in \mathbb{R}^{n\times n}$, i.e,, a $(\beta, a, \widetilde{\mathcal{O}}(\theta/\beta n^2))$-block-encoding. Then, we can create a purification of a state $\tilde{\rho}$ such that*

$$\left\| \tilde{\rho} - \frac{\exp(H)}{\operatorname{tr}(\exp(H))} \right\|_{\mathrm{tr}} \leq \theta$$

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

Provided access to a unitary that prepares a purification of a density operator, we can also construct a block-encoding of it. This is formalized in the following lemma from [23], which was based on ideas found in [46, Corollary 9].

Lemma 7 (Lemma 6.4.4 in [23]). *(Block-encoding of a (subnormalized) density operator) Let G be a $(w+a)$ unitary which on the input state $|0\rangle^w |0\rangle^a$ prepares a purification $|\underline{\rho}\rangle$ of the subnormalized w-qubit density operator ρ. Then we can implement a $(1, w+a, 0)$-block-encoding of ρ with a single use of G and its inverse and with $w+1$ two-qubit gates.*

We are now in a position to define a trace estimator using the quantum operator input model.

Lemma 8 (Lemma 4.18 in [62]). *Let ρ be an n-dimensional quantum state and U an $(\alpha, a, \theta/2)$-block-encoding of a matrix $A \in \mathbb{R}^{n\times n}$ with $\|A\| \leq 1$. A trace estimator for $\operatorname{tr}(A\rho)$ with bias at most θ and $\sigma = \mathcal{O}(1)$ can be implemented using $\widetilde{\mathcal{O}}(\alpha)$ uses of U and $U^\dagger$ and $\widetilde{\mathcal{O}}_{\frac{1}{\theta}}(\alpha)$ elementary operations.*

2.1.5 Computational complexity

When discussing the computational complexity of quantum algorithms we normally express the cost in terms of the number of calls to some input oracle. Unless otherwise specified, the gate complexity is at most a poly-logarithmic factor larger than the stated oracle complexity. The meaning of "input oracle access" depends on the input model:

- For the sparse-oracle access model, it refers to a query to the oracle describing $C/\|C\|_F$.
- For the QRAM model, it refers to the number of accesses to QRAM. A QRAM of size $\mathcal{O}\left(ns\log^2(n)\right)$ is sufficient for our algorithms, and in particular, we only need classical write access to the QRAM, i.e., we do not write in superposition.

It is straightforward to translate each of these oracle costs into a running time in the standard gate model without QRAM, by considering the cost of implementing each oracle.

3 Hamiltonian Updates

In this section, we present the algorithm from [11] and relevant results required to prove its convergence and analyze its cost.

3.1 Convex Feasibility Problems

In order to avoid any normalization issues for the problems that arise over the course of our IR scheme, we deviate slightly from [11] and renormalize the problem (3) using the Frobenius norm of the cost matrix

rather than use its operator norm:

$$\begin{aligned} \text{find} \quad & X \\ \text{subject to} \quad & \operatorname{tr}\left(\frac{C}{\|C\|_F} X\right) \geq \gamma - \epsilon \\ & \sum_{i \in [n]} \left| \langle i|X|i\rangle - \frac{1}{n} \right| \leq \epsilon \\ & \operatorname{tr}(X) = 1, \quad X \succeq 0. \end{aligned} \tag{5}$$

The relaxed renormalized SDO problem (5) is a specific example of the convex optimization problem

$$\begin{aligned} \max \quad & f(X) \\ \text{subject to} \quad & X \in \mathcal{P}_1 \cap \mathcal{P}_2 \cap \cdots \cap \mathcal{P}_m, \\ & \operatorname{tr}(X) = 1, \ X \succeq 0, \end{aligned} \tag{6}$$

where $\mathcal{P}_1, \ldots, \mathcal{P}_m$ are convex sets.

In this context, the trace constraint enforces normalization, but also allows us to obtain a bound on the optimal objective value. Letting $\widetilde{C} = C\|C\|_F^{-1}$ and invoking the tracial matrix Hölder inequality [9], it follows that any X^* that solves (6) satisfies the following relation:

$$\left|\operatorname{tr}(\widetilde{C}X^*)\right| \leq \|\widetilde{C}\|\|X^*\|_{\mathrm{tr}} = \|\widetilde{C}\|.$$

It is well known in the optimization literature that performing binary search over the range of values

$$\gamma \in \left[-\|\widetilde{C}\|, \|\widetilde{C}\|\right] \subseteq [-1, 1]$$

that the objective can take reduces the task of solving (6) to solving a sequence of feasibility problems of the form:

$$\begin{aligned} \text{find} \quad & X \in \mathcal{S}_+^n \cap \{X : \operatorname{tr}(X) = 1\} \\ \text{subject to} \quad & \operatorname{tr}(\widetilde{C}X) \geq \gamma \\ & X \in \mathcal{P}_1 \cap \mathcal{P}_2 \cap \cdots \cap \mathcal{P}_m. \end{aligned} \tag{7}$$

In particular, $\log(\|\widetilde{C}\|\epsilon^{-1}) = \mathcal{O}\left(\log(\epsilon^{-1})\right)$ queries to (7) are sufficient to estimate the optimal objective value of of (6) up to additive error ϵ.

3.2 Solving Convex Feasibility Problems via Hamiltonian Updates

Hamiltonian Updates (HU) is a meta-algorithm for solving convex feasibility problems of the form (7), adapted from the work of Tsdua, Rätsch and Warmuth [61] as well as [5, 10, 33, 43]. At a high level, HU can be viewed as a mirror descent algorithm [48, 49] with the von Neumann entropy as the mirror map.[5] In each iteration, the method uses certain subroutines to test ϵ-closeness to convex sets $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$, which we formally define next.

Definition 6 (Definition 2.1 in [11]). *Let $\mathcal{P} \subset \{X \in \mathcal{S}_+^n : \operatorname{tr}(X) = 1\}$ be a closed, convex subset of quantum states, and $\widetilde{\mathcal{P}} \subset \{X \in \mathbb{C}^{n \times n} : X = X^\dagger, \|X\| \leq 1\}$ be a closed, convex subset of observables of operator norm at most 1. For $\epsilon > 0$, an ϵ-separation oracle with respect to $\widetilde{\mathcal{P}}$ is a subroutine that either accepts a state ρ (in the sense that observables from $\widetilde{\mathcal{P}}$ cannot distinguish ρ from the elements of $\mathcal{P}$), or provides a normal vector (in the matrix space) P of a hyperplane that separates ρ from the set $\mathcal{P}$ using a test from $\widetilde{\mathcal{P}}$:*

$$O_{\mathcal{P},\epsilon}(\rho) = \begin{cases} \textit{accept } \rho & \textit{if } \min_{Y \in \mathcal{P}} \max_{P \in \widetilde{\mathcal{P}}} \operatorname{tr}(P(\rho - Y)) \leq \epsilon, \\ \textit{output } P \in \widetilde{\mathcal{P}} \textit{ s.t. } \operatorname{tr}(P(\rho - Y)) \geq \frac{\epsilon}{2} \textit{ for all } Y \in \mathcal{P} & \textit{otherwise.} \end{cases}$$

[5] Allen-Zhu and Orecchia show how MMWU algorithms can be derived from mirror descent in [1, Appendix A.2].

The authors in [11] point out that the above oracle construction is well defined, as we can always choose some hyperplane $P \in \widetilde{\mathcal{P}}$ such that

$$\operatorname{tr}\left(P(\rho - Y)\right) \geq \frac{\epsilon}{2},$$

holds for all $Y \in \mathcal{P}$ whenever

$$\min_{Y \in \mathcal{P}} \max_{P \in \widetilde{\mathcal{P}}} \operatorname{tr}(P(\rho - Y)) > \epsilon.$$

From Sion's min-max theorem [58], it follows that

$$\max_{P \in \widetilde{\mathcal{P}}} \min_{Y \in \mathcal{P}} \operatorname{tr}(P(\rho - Y)) = \min_{Y \in \mathcal{P}} \max_{P \in \widetilde{\mathcal{P}}} \operatorname{tr}(P(\rho - Y)) > \epsilon,$$

and hence there exists a hyperplane which separates ρ from $\mathcal{P}$ by ϵ. By relaxing the requirement to $\frac{\epsilon}{2}$-separation, the algorithm is able to reconcile with the errors that result from approximating quantities computed with ρ, or estimating its entries.

The Hamiltonian Updates (HU) algorithm of Brandão et al. [11] is provided in full detail in Algorithm 1. The algorithm takes as input the precision parameter ϵ, and m ϵ-separation oracles $O_{1,\epsilon}, O_{2,\epsilon}, \ldots, O_{m,\epsilon}$. In the initialization steps, the starting point is defined to be the maximally mixed state $\rho \leftarrow n^{-1} I$. This is critical to ensuring the convergence of mirror descent-based approaches such as Algorithm 1 and the works in [5, 10, 33, 43, 61]; initialization to the maximally mixed state ensures that the quantum relative entropy between any feasible state and the initial state is bounded by $\log(n)$ (see, e.g., Theorem 11.8 pt. 2 [54]), and is reduced at every iteration. Consequently, Algorithm 1 terminates in a finite number of iterations.

As noted in [11], how we define $\widetilde{\mathcal{P}}$ determines the number of closeness conditions that need to be tested. By using the Gibbs state change of variables, we do not need to test if our candidate solution is trace normalized or positive semidefinite; any Gibbs state

$$\rho_H = \frac{\exp(-H)}{\operatorname{tr}(\exp(-H))}$$

is an element of the set $\{X \in \mathcal{S}_+^n : \operatorname{tr}(X) = 1\}$ by definition. Our task therefore reduces to finding a $\log(n)$-qubit mixed state ρ which is ϵ-close to the convex sets $\mathcal{P}_i$ that arise from any other constraints included in the feasibility problem. At each iteration, ϵ-closeness is tested by querying ϵ-separation oracles which are constructed using observables in $\widetilde{\mathcal{P}}_i$. If each of our oracles accepts the candidate state, the algorithm terminates and reports (ρ, H) as an ϵ-precise solution. Otherwise, upon detecting infeasibility the matrix exponent is updated to penalize the infeasible directions using the rule

$$H \leftarrow H + \frac{\epsilon}{16} P,$$

where P is a normal vector in the matrix space of a hyperplane that witnesses infeasibility.

Algorithm 1 Hamiltonian Updates for Convex Feasibility Problems

Input: Error tolerance $\epsilon \in (0, 1)$, query access to m ϵ-separation oracles $O_{1,\epsilon}(\cdot), \ldots, O_{m,\epsilon}(\cdot)$
Initialize $\rho \leftarrow n^{-1} I$ and $H \leftarrow \mathbf{0}^{n \times n}$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, m$ **do**
    if $O_{i,\epsilon}(\rho) = P$ **then**
      $H \leftarrow H + \frac{\epsilon}{16} P$
      $\rho \leftarrow \frac{\exp(-H)}{\operatorname{tr}(\exp(-H))}$
      break
    end
  end
  return (ρ, H) and **exit**
end

The following result establishes the iteration complexity of Algorithm 1.

Theorem 3 (Theorem 2.1 in [11]). *Algorithm 1 requires at most* $T = \lceil 64 \log(n)\epsilon^{-2} \rceil + 1$ *iterations to certify that* (7) *is infeasible or output a state* ρ *satisfying*

$$\text{for all } 1 \leq i \leq m: \quad \max_{P_i \in \widetilde{\mathcal{P}}_i} \min_{Y_i \in \mathcal{P}_i} \operatorname{tr}(P_i(\rho - Y_i)) \leq \epsilon.$$

Note that Theorem 3 applies to *any* convex feasibility problem (on density operators, i.e., trace-normalized positive semidefinite matrices) for which we have separation oracles as outlined in Definition 6. This is crucial for the development of an iterative refinement scheme.

There is an important distinction with respect to output across the models of computation we study. A classical implementation of Algorithm 1 outputs an explicit description of an ϵ-precise solution ρ^* to (5) and its associated Hamiltonian H^*, whereas a quantum implementation reports a real valued vector $y \in \mathbb{R}^2$ along with a diagonal matrix D (with $\|D\| \leq 1$) such that $H^* = y_1\widetilde{C} + y_2 D$. The vector $y = (y_1, y_2)^\top$ is the *state preparation pair* of ρ^*, in particular:

$$\rho^* = \frac{\exp\left(-\left(y_1\widetilde{C} + y_2 D\right)\right)}{\operatorname{tr}\left[\exp\left(-\left(y_1\widetilde{C} + y_2 D\right)\right)\right]},$$

and we refer to this type of output as a *state preparation pair description* of ρ. This choice of output is used in all quantum SDO solvers based on Gibbs sampling techniques (see, e.g., [10, 11, 12, 64, 65]), and is motivated by the fact that it is difficult to develop quantum algorithms that are substantially faster than classical algorithms if we still have to output each entry of the solution (an $n \times n$ matrix).

The Gibbs sampling approaches that we apply later exhibit a cost that depends on a norm bound for y. Observe that we initialize y to the all zeros vector of appropriate dimension, and in every iteration, at most one entry of y changes by a magnitude of $\frac{\epsilon}{16}$ (specifically, an entry y_i, where the oracle $O_{i,\epsilon}$ has detected infeasibility). As a consequence, the vector y satisfies the inequality

$$\left\| y^{(t+1)} - y^{(t)} \right\| \leq \frac{\epsilon}{16} \tag{8}$$

for each iteration t. In view of the iteration bound for Algorithm 1 provided in Theorem 3, it is easy to see that for any y obtained from Algorithm 1 we have

$$\|y\|_1 \leq \lceil 64 \log(n)\epsilon^{-2} \rceil \left\| y^{(t+1)} - y^{(t)} \right\| \leq \lceil 64 \log(n)\epsilon^{-2} \rceil \frac{\epsilon}{16} \leq 4 \log(n)\epsilon^{-1}. \tag{9}$$

To instantiate the algorithm to solve problem (3) we need to choose the sets $\mathcal{P}_i$, and provide separation oracles for them. This is what we do in the following section.

3.2.1 Oracle Construction

The goal of Hamiltonian Updates is to solve, for fixed $\gamma \in [-1, 1]$, the following feasibility problem:

$$\begin{aligned} \text{find} \quad & \rho \in \{X \in \mathcal{S}_+^n : \operatorname{tr}(X) = 1\} \cap \mathcal{C}_\gamma \cap \mathcal{D}_n \\ \text{where} \quad & \mathcal{C}_\gamma = \left\{ X : \operatorname{tr}\left(\widetilde{C} X\right) \geq \gamma \right\}, \\ & \mathcal{D}_n = \left\{ X : \langle i|X|i\rangle = \frac{1}{n}, i \in [n] \right\}. \end{aligned} \tag{10}$$

One can observe that the set $\mathcal{C}_\gamma$ constitutes a halfspace, while $\mathcal{D}_n$ is an affine space of codimension n. The sets of observables for $\mathcal{C}_\gamma$ and $\mathcal{D}_n$ are given by $\widetilde{\mathcal{C}}_\gamma$ and $\widetilde{\mathcal{D}}_n$ respectively, with

$$\widetilde{\mathcal{C}}_\gamma = \{-\widetilde{C}\}, \text{ and } \widetilde{\mathcal{D}}_n = \{D \in \mathbb{R}^{n \times n} : \|D\| \leq 1, \ D \text{ is diagonal}\}.$$

As noted in [11], it follows

$$\max_{P\in\widetilde{\mathcal{C}}_\gamma}\min_{Y\in\mathcal{C}_\gamma}\operatorname{tr}(P(\rho-Y))\le\epsilon \iff -\operatorname{tr}\left(\widetilde{C}(\rho-Y)\right)\le\epsilon \quad \text{for some } Y\in\mathcal{C}_\gamma,$$

which in turn implies $\operatorname{tr}(\widetilde{C}\rho)\ge\gamma-\epsilon$.

Given the structure of $\mathcal{C}_\gamma$ and $\mathcal{D}_n$, the authors in [11] suggest the following two ϵ-separation oracles:

$O_{\mathcal{C}_\gamma}$: compute an approximation $\tilde{c}$ of $\operatorname{tr}\left(\widetilde{C}\rho\right)$ up to additive error $\frac{\epsilon}{4}$. Check if $\tilde{c}\ge\gamma-\frac{3\epsilon}{4}$ and output $P=-\widetilde{C}$ if the inequality is violated.

$O_{\mathcal{D}_n}$: compute an approximation $\tilde{p}\in\mathbb{R}^n$ of $p_i=\langle i|\rho|i\rangle$ satisfying $\sum_{i=1}^{n}|p_i-\tilde{p}_i|\le\frac{\epsilon}{4}$.
Check if $\sum_{i=1}^{n}\left|\tilde{p}_i-\frac{1}{n}\right|\le\frac{3\epsilon}{4}$ and output $P=\sum_{i=1}^{n}\left(\mathbb{I}\left\{\tilde{p}_i>\frac{1}{n}\right\}-\mathbb{I}\left\{\tilde{p}_i<\frac{1}{n}\right\}\right)|i\rangle\langle i|$ if the inequality is violated.

For any given

$$\rho_H=\frac{\exp(-H)}{\operatorname{tr}(\exp(-H))},$$

the required separation oracles are straightforward to implement on a classical computer that has access to ρ_H. Thus, classically we only need to prepare ρ_H once and store it to build the separation oracles. The next result from [11] establishes that computing an $\mathcal{O}(\log(n)\epsilon^{-1})$-degree Taylor series suffices to produce accurate approximations.

Lemma 9 (Lemma 3.2 in [11])**.** *Fix a Hermitian $n\times n$ matrix H, an accuracy ϵ, and let ℓ be the smallest even number satisfying $(\ell+1)(\log(\ell+1)-1)\ge 2\|H\|+\log(n)+\log\left(\frac{1}{\epsilon}\right)$. Then, the truncated matrix exponential $T_\ell=\sum_{k=0}^{\ell}\frac{1}{k!}(-H)^k$ satisfies*

$$\left\|\frac{\exp(-H)}{\operatorname{tr}\left(\exp(-H)\right)}-\frac{T_\ell}{\operatorname{tr}(T_\ell)}\right\|_{\mathrm{tr}}\le\epsilon.$$

The task of implementing our separation oracles and testing feasibility on a quantum computer reduces to preparing Gibbs states [11], which are used to test closeness to the sets $\mathcal{C}_\gamma$ and $\mathcal{D}_n$ via quantum measurements. While in Lemma 9 we bound the number of required Taylor series steps for computing ρ via a matrix exponential, in the quantum case we bound the number of copies of ρ required to estimate its diagonal entries and expectation values $\operatorname{tr}(A\rho)$.

Lemma 10. *Fix $\epsilon\in(0,1)$. Let ρ be a $\log(n)$-qubit quantum state and U a $(1,\log(n)+2,\epsilon/(2n))$-block-encoding of $\widetilde{C}=C/\|C\|_F^{-1}$. Then, we can implement the oracle $O_{\mathcal{C}_\gamma}$ on a quantum computer given access to $\mathcal{O}(\epsilon^{-1})$ copies of a state that is an $\frac{\epsilon}{8}$-approximation of the input state ρ in trace distance and $\mathcal{O}(\epsilon^{-1})$ applications of U and $U^\dagger$. The oracle $O_{\mathcal{D}_n}$ can be implemented using $\mathcal{O}(n\epsilon^{-2})$ $\frac{\epsilon}{8}$-approximate copies of the input, and the classical post-processing time needed to implement the oracle is $\mathcal{O}(n\epsilon^{-2})$.*

Proof. First, note that we can obtain an estimate $\tilde{p}$ of the diagonal elements of ρ whose total variation distance from p is no more than $\frac{\epsilon}{8}$ using $\widetilde{\mathcal{O}}_n\left(n\epsilon^{-2}\right)$ copies of ρ to measure ρ in the computational basis. Further, provided accesses to ρ and a $(1,\log(n)+2,\epsilon/(2n))$-block-encoding U of $\widetilde{C}$, by Lemma 8, a trace estimator for $\operatorname{tr}\left(\widetilde{C}\rho\right)$ with bias at most $\frac{\epsilon}{n}$ can be implemented using $\widetilde{\mathcal{O}}(1)$ uses of U and $U^\dagger$ and $\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}(1)$ elementary operations. From here, applying amplitude estimation using $\mathcal{O}(\epsilon^{-1})$ quantum samples (i.e., state preparation unitaries) from the trace estimator to suffice to compute an approximation $\operatorname{tr}\left(\widetilde{C}\rho\right)$ up to additive $\frac{\epsilon}{8}$ to implement $O_{\mathcal{C}_\gamma}$. The rest of the proof exactly follows the proof of [11, Lemma 3.3]. □

We remark that multidimensional phase estimation techniques from [63] could improve the dependence on ϵ^{-1} for estimating the diagonal elements of ρ to linear, which is a factor ϵ^{-1} better than a näive application of computational basis measurements. However, in the context of the iterative refinement scheme we present later, the improvement would only reduce the amount of constant overhead in the overall running time, and multidimensional phase estimation has a larger gate complexity (which can be reduced with QRAM). There are also numerous ways to prepare Gibbs states using a quantum computer [17, 21, 38, 57, 64, 65, 67]. Following [11], we utilize the Gibbs sampler from [57] when working with the sparse-access input model, and for the QRAM input model we consider Gibbs sampling techniques introduced in [64].

3.3 Complexity

Having understood the cost of constructing the oracles in both the classical and quantum settings, we are now in a position to analyze the complexity associated with using Algorithm 1 to obtain solutions to (5) and approximations to (3). Relevant to this discussion is the following result, which imposes precision requirements on solving (3) to an additive error of the order $\mathcal{O}\left(n\|C\|_F\epsilon\right)$ using Algorithm 1.

Proposition 5 (Proposition 3.1 in [11])**.** *Let ρ be an ϵ^4-accurate solution to the relaxed SDO problem* (5) *with input matrix C. Let $\gamma_{\epsilon^4} = \operatorname{tr}(C\rho)$ be the value attained by ρ. Then, there is a quantum state ρ^* at trace distance $\mathcal{O}(\epsilon)$ of ρ such that $n\rho^*$ is a feasible point of SDO problem* (3)*. In particular*

$$|\gamma_{\epsilon^4} n\|C\|_F - \operatorname{tr}(n\rho^* C)| = \mathcal{O}\left(n\|C\|_F\epsilon\right).$$

Moreover, it is possible to construct ρ^* in time $\mathcal{O}(n^2)$ given the entries of ρ.

We do not provide a proof of this result here, as later we will provide an improved approximation guarantee and a proof of the improved statement.

3.3.1 Classical running time

Using Lemma 9 in combination with Theorem 3, we can bound the running time required to solve (5) to additive error ϵ using a classical implementation of Algorithm 1.

Proposition 6. *Suppose that C has row sparsity s. Then, the classical cost of solving* (5) *up to additive error ϵ using Algorithm 1 is $\mathcal{O}\left(\min\{n^2 s, n^\omega\}\log^2(n)\epsilon^{-3}\right)$.*

Proof. The result follows directly from the proof of Corollary 3.1 in [11], but we repeat the argument here for completeness.

First, observe that over the course of the iterations $t = 0, \ldots, T$, the operator norms $\|H^{(t)}\|$ do not become prohibitively large. This follows from initializing $H^{(0)} = \mathbf{0}^{n\times n}$, and that by (8), the inequality

$$\left\|H^{(t+1)} - H^{(t)}\right\| \leq \frac{\epsilon}{16}\left\|P^{(t)}\right\| \leq \frac{\epsilon}{16}$$

holds for all t. By Theorem 3, Algorithm 1 requires at most $T = \lceil 64\log(n)\epsilon^{-2}\rceil$ iterations, which implies $\|H^{(t)}\| \leq 4\log(n)\epsilon^{-1}$ for all t.

By Lemma 9, it suffices to compute $\mathcal{O}(\log(n)\epsilon^{-1})$ steps of the Taylor series corresponding to $\exp(-H^{(t)})$ in order to obtain a matrix $\tilde{\rho}^{(t)}$ that is at most a trace distance of $\frac{\epsilon}{4}$ from $\rho^{(t)}$. Moreover, given that $H^{(t)}$ is defined as a linear combination of $\widetilde{C}$ with a diagonal matrix, matrix multiplication involving $H^{(t)}$ can be carried out in $\mathcal{O}(\min\{n^2 s, n^\omega\})$ arithmetic operations. Given classical access to $\tilde{\rho}^{(t)}$, the diagonal constraints comprising $\mathcal{D}_n$ can be checked in time $\mathcal{O}(n)$, whereas computing $\operatorname{tr}\left(\widetilde{C}\tilde{\rho}^{(t)}\right)$ requires $\mathcal{O}(ns)$ arithmetic operations. Thus, the dominant operation at each iteration is computing the matrix exponential and the classical per-iteration cost of Algorithm 1 is given by

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\}\log(n)\epsilon^{-1}\right).$$

Taking into account the iteration bound $\mathcal{O}(\log(n)\epsilon^{-2})$ provided in Theorem 3, we arrive at an overall running time of

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\}\log^2(n)\epsilon^{-3}\right).$$

The proof is complete. □

The next corollary from [11] follows from Proposition 5 in the context of the previous result, and provides the overall running time of Algorithm 1 to solve (3) to additive error $\mathcal{O}\left(n\|C\|_F\epsilon\right)$ in the classical setting.

Corollary 1. *Suppose that C has row-sparsity s. Then, the classical cost of solving (3) up to an additive error $\mathcal{O}\left(n\|C\|_F\epsilon\right)$ using Algorithm 1 is $\mathcal{O}\left(\min\{n^2 s, n^\omega\}\log^2(n)\epsilon^{-12}\right)$.*

Proof. By Proposition 6, Algorithm 1 requires time

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\}\log^2(n)\tilde{\epsilon}^{-3}\right),$$

to solve (5) up to additive error $\tilde{\epsilon}$. In order to satisfy the approximation guarantee for (3) given in Proposition 5, it suffices to solve (5) to error $\tilde{\epsilon} = \epsilon^4$. Plugging in this value for the precision parameter, the total cost required to solve (3) up to an additive error $\mathcal{O}\left(n\|C\|_F\epsilon\right)$ using Algorithm 1 is

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\}\log^2(n)\tilde{\epsilon}^{-3}\right) = \mathcal{O}\left(\min\{n^2 s, n^\omega\}\log^2(n)(\epsilon^4)^{-3}\right) = \mathcal{O}\left(\min\{n^2 s, n^\omega\}\log^2(n)\epsilon^{-12}\right).$$

□

3.3.2 Quantum running time

Combining the sampling requirements provided in Lemma 10 with the cost of preparing a single Gibbs state and the iteration bound from Theorem 3 gives the complexity of Algorithm 1 when run on a quantum computer. However, Gibbs samplers based on the block-encoding framework depend only poly-logarithmically on the inverse precision, therefore they are exponentially faster (in the parameter ϵ^{-1}) compared to the Gibbs sampling algorithm from [57] utilized in [11]. It thus makes sense to analyze the running time in the more efficient model. This will require an efficient data structure for storing y so that we can efficiently prepare linear combinations of block-encodings.

Lemma 11 (Lemma 15 in [64])**.** *There is a data structure that can store an m-dimensional χ-sparse vector y with θ-precision using a QRAM of size $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(\chi)$. Furthermore:*

- *Given a classical $\mathcal{O}(1)$-sparse vector, adding it to the stored vector has classical cost $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(1)$.*
- *Given that $\beta \geq \|y\|_1$, we can implement a (symmetric) $(\beta, \widetilde{\mathcal{O}}_{\frac{m}{\theta}}(1), \theta)$-state preparation pair for y with $\widetilde{\mathcal{O}}_{\frac{m}{\theta}}(1)$ queries to the QRAM.*

Corollary 2 (Corollary 16 in [64])**.** *Suppose $A_1, \ldots, A_m$ are Hermitian matrices with operator norm at most 1, and that $y \in \mathbb{R}^m$ satisfies $\|y\|_1 \leq \beta$. Having access to the above data structure for y, we can prepare one copy of the Gibbs state*

$$\rho = \frac{\exp\left(-\sum_{i=1}^m y_i A_i\right)}{\operatorname{tr}\left(\exp\left(-\sum_{i=1}^m y_i A_i\right)\right)}$$

using $\widetilde{\mathcal{O}}_\theta(\sqrt{n}\alpha\beta)$ accesses to the data structure for y and block-encodings of $A_1, \ldots, A_m$.

We can now use Corollary 2 in combination with results from Sections 2.1.3 and 2.1.4 to establish the running time of Algorithm 1 in the QRAM input model.

Proposition 7. *Let $\widetilde{C} = C\|C\|_F^{-1} \in \mathcal{S}^n$ be stored in QRAM. Then, the complexity of solving (5) up to additive error ϵ with Algorithm 1 using the QRAM input model is*

$$\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(n^{1.5}\epsilon^{-5}\right).$$

Here, the complexity corresponds to the number of accesses to the QRAM.

Proof. Given that $\widetilde{C}$ is stored in QRAM, Lemma 3*(ii)* asserts that when constructing a block-encoding of $\widetilde{C}$, one can set the subnormalization factor to be $\alpha_C = \left\|\widetilde{C}\right\|_F = 1$. Hence, one can construct a $(1, \log(n) + 2, \epsilon/(2n))$-block-encoding of $\widetilde{C}$ in time $\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}(1)$.

Next, recall that in iteration $t \in [T]$ of Algorithm 1, our Hamiltonian is defined as

$$H^{(t)} = y_1^{(t)}\widetilde{C} + y_2^{(t)}D^{(t)},$$

where $D^{(t)}$ is a diagonal matrix with the diagonal entries taking value -1, 0 or 1. The diagonal elements of D change in each iteration, and therefore, a new D must be block-encoded in each iteration. For this, we use the QRAM model described in Section 2.1.2, which allows for insertions to be made in time $\widetilde{\mathcal{O}}_n(1)$ to keep the cost of this step negligible. Provided a classical description of D, we can store D in the QRAM in time $\mathcal{O}(n\log(n))$. Applying Lemma 4, a $(1, \log(n) + 3, \epsilon)$-block-encoding of $D^{(t)}$ can be constructed in time $\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}(1)$.

In an earlier discussion we saw that any y obtained from a call to Algorithm 1 will satisfy $\|y\|_1 = \widetilde{\mathcal{O}}_n(\epsilon^{-1})$ if we call Algorithm 1 using precision ϵ (see, e.g., equation (9)). Hence, an application of Corollary 2 with $\beta = \widetilde{\mathcal{O}}_n(\epsilon^{-1})$ implies that we can prepare one copy of our Gibbs state using

$$\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(\sqrt{n}\alpha\epsilon^{-1}\right)$$

accesses to the data structure for y and the block-encodings of $\widetilde{C}$ and D, where α is defined as the maximum over the subnormalization factors used to block-encode $\widetilde{C}$ and D. Since $\alpha = \max\{\alpha_C, \alpha_D\} = 1$, it follows

$$\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(\sqrt{n}\alpha\epsilon^{-1}\right) = \widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(\sqrt{n}\epsilon^{-1}\right).$$

Now, one can see from Lemma 10 that the cost of constructing $O_{\mathcal{D}_n}$ dominates that of constructing $O_{\mathcal{C}_\gamma}$. Noting that $O_{\mathcal{D}_n}$ can be implemented using $\mathcal{O}(n\epsilon^{-2})$ copies of a state that is an $\frac{\epsilon}{8}$-approximation of the input state ρ in trace distance and its inverse, the per-iteration cost of Algorithm 1 in the QRAM input model is given by

$$\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(n^{1.5}\epsilon^{-3}\right).$$

Factoring in the iteration bound of $\widetilde{\mathcal{O}}_n(\epsilon^{-2})$ from Theorem 3, it follows that when provided access to QRAM, Algorithm 1 solves (5) up to additive error ϵ using

$$\mathcal{T}_{HU}^{\text{quantum}} = \widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(n^{1.5}\epsilon^{-5}\right)$$

accesses to the QRAM. The proof is complete. □

Corollary 3. *Let $\widetilde{C} \in \mathcal{S}^n$ be stored in QRAM. Then, the complexity of solving* (3) *up to additive error $\mathcal{O}(n\|C\|_F\epsilon)$ with Algorithm 1 using the QRAM input model is*

$$\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(n^{1.5}\epsilon^{-20}\right).$$

Here, the complexity corresponds to the number of accesses to the QRAM.

Proof. By Proposition 7, Algorithm 1 requires

$$\widetilde{\mathcal{O}}_{\frac{n}{\tilde{\epsilon}}}\left(n^{1.5}\tilde{\epsilon}^{-5}\right),$$

accesses to the QRAM to solve (5) up to additive error $\tilde{\epsilon}$. In order to satisfy the approximation guarantee for (3) given in Proposition 5, it suffices to solve (5) to error $\tilde{\epsilon} = \epsilon^4$. Plugging in this value for the precision parameter, the total cost required to solve (3) up to an additive error $\mathcal{O}\left(n\|C\|_F\epsilon\right)$ using Algorithm 1 is

$$\widetilde{\mathcal{O}}_{\frac{n}{\tilde{\epsilon}}}\left(n^{1.5}\tilde{\epsilon}^{-5}\right) = \widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(n^{1.5}(\epsilon^4)^{-5}\right) = \widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(n^{1.5}\epsilon^{-20}\right).$$

The proof is complete. □

Corollary 3 establishes that utilizing Gibbs samplers and trace estimators based on the block-encoding framework for our oracle construction in Algorithm 1 leads to an

$$\mathcal{O}\left(\sqrt{s}^{1+o(1)}\epsilon^{-8+o(1)}\exp\left(1.6\sqrt{\log(\epsilon^{-4})}\right)\right)$$

speedup over the running time result provided in [11, Corollary 3.2] when applied to solving (3). Yet, the costly accuracy requirements for the rounding procedure (see, e.g., Proposition 5) lead to a prohibitive scaling in the inverse precision for the overall running time. Given the advantageous dependence on the dimension, as compared to classical algorithms, we study how to improve the dependence on the precision parameter. This is discussed next.

4 Iterative Refinement for SDO approximations of QUBOs

In this section, we introduce an iterative refinement method for obtaining accurate solutions to the renormalized relaxed SDO problem (5), that at a high level can be viewed as solving a series of problems related to the *feasibility problem* (10) associated with (5). We then discuss how to test ϵ-closeness to the convex sets which comprise the feasible regions of the intermediate refining problems before presenting our algorithm in full detail. We conclude the section by proving our algorithm's correctness and iteration complexity, and use these results to provide an improved approximation guarantee.

4.1 The refining problem

To develop an iterative refinement scheme for (5), we need to design a problem whose solution can be used to improve the quality of solutions to (5). Suppose we run Algorithm 1 and obtain an ϵ-precise solution $\tilde{\rho}$ to (5). Letting $\tilde{\gamma} = \operatorname{tr}\left(\widetilde{C}\tilde{\rho}\right)$, $\tilde{\rho}$ must satisfy

$$\begin{aligned}\operatorname{tr}\left(\widetilde{C}\tilde{\rho}\right) = \tilde{\gamma} \geq \gamma - \epsilon,\\ \sum_{i=1}^{n}\left|\langle i|\tilde{\rho}|i\rangle - \frac{1}{n}\right| \leq \epsilon.\end{aligned}$$

In *refining* our solution to (5), we should aim to reduce the total variation distance from the distribution along the diagonal elements of our solution to the uniform distribution, while also improving the precision to which the optimal objective value is approximated. Thus, an improved solution ρ' should obey

$$\begin{aligned}\operatorname{tr}\left(\widetilde{C}\rho'\right) \geq \gamma - \epsilon',\\ \sum_{i=1}^{n}\left|\langle i|\rho'|i\rangle - \frac{1}{n}\right| \leq \epsilon',\end{aligned}$$

with $\epsilon' < \epsilon$. The basic idea behind constructing the refining problem is to use our current solution $\tilde{\rho}$ to first shift the renormalized relaxed SDO problem (5) to the origin, and then scale the shifted problem back to the domain of the original problem. In particular, we solve a series of problems related to the feasibility problem (10).

Let $\varepsilon \in \mathbb{R}^n$ be a vector whose elements are the residuals along the diagonal $\varepsilon_i = \tilde{\rho}_{ii} - \frac{1}{n}$ for $i \in [n]$, and $\eta \geq 1$ to be a scalar defined as

$$\eta = \frac{1}{\max\left\{\gamma - \operatorname{tr}\left(\widetilde{C}\tilde{\rho}\right), \sum_{i=1}^{n}|\varepsilon_i|\right\}} = \frac{1}{\max\left\{\gamma - \operatorname{tr}\left(\widetilde{C}\tilde{\rho}\right), \left\|\sum_{i\in[n]}\langle i|\,\tilde{\rho}\,|i\rangle|i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}}\right\}}.$$

Using these quantities, the *refining problem* is given by:

$$\begin{aligned} \text{find} \quad & \rho^r \in \{X \in \mathcal{S}_+^n : \operatorname{tr}(X) = 1\} \cap \mathcal{C}_{\eta(\gamma-\tilde{\gamma})} \cap \mathcal{D}_{\eta\varepsilon} \\ \text{where} \quad & \mathcal{C}_{\eta(\gamma-\tilde{\gamma})} = \left\{X : \operatorname{tr}\left(\widetilde{C}\,(Q \circ X)\right) \geq \eta(\gamma - \tilde{\gamma})\right\}, \\ & \mathcal{D}_{\eta\varepsilon} = \{X : \langle i|X|i\rangle = \eta|\varepsilon_i|,\ \forall i \in [n]\}, \end{aligned} \tag{11}$$

where $Q \in \mathcal{S}^n$ is a matrix whose diagonal elements are chosen such that for any $X \in \mathcal{D}_{\eta\varepsilon}$, we have

$$(Q \circ X)_{ii} = \operatorname{sign}(-\varepsilon_i)\eta|\varepsilon_i|$$

for $i \in [n]$. Further details and requirements on the structure of Q are specified later in this section. We refer to solutions ρ^r to (11) as *refining solutions*, which we use to update our current solution $\tilde{\rho}$ to (5).

The set $\mathcal{D}_{\eta\varepsilon}$ is comprised of the diagonal constraints

$$\langle i|X|i\rangle = \eta|\varepsilon_i|, \quad \forall i \in [n],$$

and similar to $\mathcal{D}_n$, is an affine space with codimension n. Our use of the absolute value function of the residuals and scaling by η ensures the viability of applying Gibbs sampling techniques to solve the refining problem (11); the diagonal terms of any density matrix must be nonnegative and sum to 1. Whenever

$$\sum_{i=1}^{n} |\varepsilon_i| > \gamma - \operatorname{tr}\left(\widetilde{C}\tilde{\rho}\right),$$

then $\eta\|\varepsilon\|_1 = 1$, and the parameter η therefore scales the shifted problem back to the space of the $\log(n)$-qubit mixed states, ensuring that any solution ρ^r to (11) is indeed a (trace normalized) Gibbs state.

On the other hand, should it be the case that

$$\sum_{i=1}^{n} |\varepsilon_i| \leq \gamma - \operatorname{tr}\left(\widetilde{C}\tilde{\rho}\right),$$

then for any $X \in \mathcal{D}_{\eta\varepsilon}$ we have $\operatorname{tr}(X) \leq 1$, rather than $\operatorname{tr}(X) = 1$. Our primal SDO oracle in Algorithm 1 solves feasibility problems in which the trace upper bound is tight, i.e., $\operatorname{tr}(X) = 1$. The authors in [64] note that this can be dealt with adding one extra variable w such that

$$\bar{\rho}^r := \begin{bmatrix} \rho^r & 0 \\ 0 & w \end{bmatrix}.$$

Then, $\operatorname{tr}(\bar{\rho}^r) = 1$ and $\bar{\rho}^r \succeq 0$ imply that $\operatorname{tr}(\rho^r) \leq 1$, and as a result we obtain an SDO problem that is equivalent to (11). Since we know exactly the amount of subnormalization, we can also get rid of the extra variable in subsequent calculations and re-scale the trace back to 1 when necessary (e.g., when combining solutions from multiple iterative refinement iterations for trace estimations). Crucially, using the input models described in Section 2.1, these modifications do not introduce more than constant overhead in the overall complexity, as the problem data in this case is simply given by

$$\overline{C} = \begin{bmatrix} \widetilde{C} & 0 \\ 0 & 0 \end{bmatrix}, \quad \overline{Q} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix},$$

with $(\overline{C}, \overline{Q}) \in \mathcal{S}^{n+1} \times \mathcal{S}^{n+1}$.

The Hadamard product $Q \circ \rho^r$ that appears in the definition of $\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}$ is required for similar reasons; properly setting Q allows us to drive the total variation distance from the distribution along the diagonal elements of our solution to the uniform distribution to zero using the solutions to the refining problem. Later, in Section 4.3 we demonstrate that this can be achieved by generating a sequence of iterates $\tilde{\rho}, \hat{\rho}$, where we obtain $\hat{\rho}$ from $\tilde{\rho}$ using the rule

$$\hat{\rho} = \tilde{\rho} + \frac{1}{\eta} Q \circ \rho^r, \tag{12}$$

with a suitable choice for Q being

$$Q = (ee^\top - I) + \operatorname{diag}(\operatorname{sign}(-\varepsilon)) = \begin{pmatrix} \operatorname{sign}(-\varepsilon_1) & 1 & \dots & 1 \\ 1 & \operatorname{sign}(-\varepsilon_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \dots & 1 & \operatorname{sign}(-\varepsilon_n) \end{pmatrix}. \tag{13}$$

Choosing Q in this manner also implies that the Hadamard product $Q \circ A$ can be carried out classically using $\mathcal{O}(n)$ arithmetic operations for any $A \in \mathbb{R}^{n\times n}$, as the element-wise products $Q_{ij}A_{ij} = A_{ij}$ for $i \neq j$. Similarly, updating Q at each iterate only requires updating its diagonal elements, an $\mathcal{O}(n)$ operation.

It is important to note that the update we propose in (12) does not preserve positive semidefiniteness in general. However, later in our analysis, we demonstrate that the eigenvalues of the updated solution $\hat{\rho}$ are only slightly negative in the worst case, i.e., $\lambda_{\min}(\hat{\rho}) \geq -\delta$ for a value $\delta > 0$ that gets progressively smaller over the course of the algorithm; one can restore positive semidefiniteness by adding δ to the diagonal elements of the final solution, and we renormalize the trace by $(1 + n\delta)$. We show that these modifications required to restore positive semidefiniteness have only a mild (in fact, constant) impact on feasibility. To this end, we will bound the eigenvalues of Q. We first state a special instance of Weyl's inequality.

Lemma 12. *Suppose that $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$ are Hermitian matrices. Then*

$$\lambda_{\min}(A + B) \geq \lambda_{\min}(A) + \lambda_{\min}(B).$$

Using the preceding lemma, the following result bounds the minimum eigenvalue of Q.

Lemma 13. *Suppose that $Q \in \mathcal{S}^n$ is defined according to Equation* (13)*. Then, $\lambda_{\min}(Q) \geq -2$.*

Proof. Let $A = (ee^\top - I)$ and $B = \operatorname{diag}(\operatorname{sign}(-\varepsilon))$, such that $Q = A + B$. Now, it can be easily seen from the definition of A that $A + I$ is an all-ones matrix of dimension n. Upon performing row-reduction (via, e.g., Guassian elimination) on A, it is trivial to observe that the resulting row-echelon form will have $n - 1$ zero rows, and as a consequence, A has the eigenvalue -1, repeated (at least) $n - 1$ times. Further, since $\operatorname{tr}(A) = 0$, the other eigenvalue is $n - 1$. Therefore, we have $\lambda_{\min}(A) \geq -1$. On the other hand, B is a diagonal matrix whose diagonal elements can take value -1, 0, or 1, from which $\lambda_{\min}(B) \geq -1$ readily follows.

Applying Lemma 12, we obtain

$$\lambda_{\min}(Q) = \lambda_{\min}(A + B) \geq \lambda_{\min}(A) + \lambda_{\min}(B) \geq -2.$$

The proof is complete. □

4.2 Oracle construction for the refining problem

In order to construct separation oracles for testing closeness to $\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}$, we rely on the following result.

Lemma 14. *Let E, F and $G \in \mathcal{S}^n$. We have*

$$\operatorname{tr}(G(E \circ F)) = \operatorname{tr}((E \circ G)F).$$

Proof. Applying Lemma 1 with $m = n$, we have

$$[(E \circ F)G]_{ii} = [(E \circ G)F]_{ii} \quad \forall i \in [n].$$

Note that we have dropped the transpose terms, as E, F and G are symmetric matrices, and hence, so are $E \circ F$ and $E \circ G$. It follows

$$\operatorname{tr}(G(E \circ F)) = \operatorname{tr}((E \circ F)G) = \sum_{i\in[n]} [(E \circ F)G]_{ii} = \sum_{i\in[n]} [(E \circ G)F]_{ii} = \operatorname{tr}((E \circ G)F).$$

□

In addition to $Q \in \mathcal{S}^n$, we also require $\max_{i,j\in[n]}\{|Q_{ij}|\} \leq 1$ to avoid any normalization issues with respect to $Q \circ \widetilde{C}$. Note that defining of Q according to equation (13) satisfies both of these properties trivially, as each of the diagonal elements are 1, 0, or -1, while the off-diagonal elements are all set to 1. This idea is formalized next.

Lemma 15. *Let $A \in \mathbb{R}^{n\times n}$ and $Q \in \mathcal{S}^n$ be matrices satisfying $\max_{i,j\in[n]}\{|Q_{ij}|\} \leq 1$ and $\|A\|_F \leq 1$. Then,*

$$\|Q \circ A\| \leq \|Q \circ A\|_F \leq 1.$$

Proof. Under the stated conditions for Q, it follows

$$\begin{aligned}\|Q \circ A\|_F^2 = \sum_{i\in[n]}\sum_{j\in[n]} \Big([Q \circ A]_{ij}\Big)^2 = \sum_{i\in[n]}\sum_{j\in[n]} (Q_{ij} \cdot A_{ij})^2 = \sum_{i\in[n]}\sum_{j\in[n]} (Q_{ij})^2 (A_{ij})^2 \\ \leq \sum_{i\in[n]}\sum_{j\in[n]} (A_{ij})^2 = \|A\|_F^2,\end{aligned}$$

and applying the square root throughout the above we obtain $\|Q \circ A\|_F \leq \|A\|_F$. From here, the result follows upon noting $\|A\|_F \leq 1$ and $\|A\| \leq \|A\|_F$ is true for any $A \in \mathbb{R}^{n\times n}$. □

Although the sets $\mathcal{C}_\gamma$ and $\mathcal{D}_n$ differ from their refining counterparts $\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}$ and $\mathcal{D}_{\eta\varepsilon}$, their dissimilarity merely affects the right hand side of the inequality defining the sets, and are thus no more difficult to construct. Just as in the case of (10), the task of obtaining separation oracles for the refining problem (11) in the quantum regime reduces to preparing many copies of Gibbs states. Likewise, these oracles can also be implemented on a classical computer, given access to ρ^r.

The similarities between (10) and (11) become transparent when we demonstrate that they are specific instances of the same problem. In particular, it is easy to see that solving (10) corresponds to solving

$$\begin{aligned}\text{find} \quad & \rho \in \{X \in \mathcal{S}_+^n : \operatorname{tr}(X) = 1\} \cap \mathcal{C}_{\eta(\gamma-\tilde{\gamma})} \cap \mathcal{D}_{\eta\varepsilon} \\ \text{where} \quad & \mathcal{C}_{\eta(\gamma-\tilde{\gamma})} = \left\{X : \operatorname{tr}\left(\widetilde{C}Q \circ X\right) \geq \eta(\gamma-\tilde{\gamma})\right\}, \\ & \mathcal{D}_{\eta\varepsilon} = \{X : \langle i|X|i\rangle = \eta|\varepsilon_i|, \ \forall i \in [n]\},\end{aligned} \tag{14}$$

with $\varepsilon_i = \frac{1}{n}$, $\eta = 1$, $Q = ee^\top$, and $\tilde{\gamma} = 0$. In view of this relationship, we can unify the oracle construction for (10) and (11) as follows:

$O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$: Compute an approximation $\tilde{c}$ of $\operatorname{tr}\left(Q \circ \widetilde{C}\rho\right)$ up to additive error $\frac{\epsilon}{4}$.
Check if $\tilde{c} \geq \eta(\gamma-\tilde{\gamma}) + \frac{3\epsilon}{4}$ and output $P = -Q \circ \widetilde{C}$ if the inequality is violated.

$O_{\mathcal{D}_{\eta\varepsilon}}$: Compute an approximation $\tilde{p} \in \mathbb{R}^n$ of $p_i = \langle i|\rho|i\rangle$ satisfying $\sum_{i\in[n]} |p_i - \tilde{p}_i| \leq \frac{\epsilon}{4}$.
Check if $\sum_{i\in[n]} |\tilde{p}_i - \eta|\varepsilon_i|| \leq \frac{3\epsilon}{4}$ and output $P = \sum_{i\in[n]} \left(\mathbb{I}\{\tilde{p}_i > \eta|\varepsilon_i|\} - \mathbb{I}\{\tilde{p}_i < \eta|\varepsilon_i|\}\right)|i\rangle\langle i|$ if the inequality is violated.

Again, the sets of observables for $\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}$ and $\mathcal{D}_{\eta\varepsilon}$ are given by

$$\widetilde{\mathcal{C}}_{\eta(\gamma-\tilde{\gamma})} = \{-Q \circ \widetilde{C}\}, \text{ and } \widetilde{\mathcal{D}}_{\eta\varepsilon} = \{D \in \mathbb{R}^{n\times n} : \|D\| \leq 1, D \text{ is diagonal}\}.$$

Although these observations are straightforward, they justify our use of Algorithm 1 as a semidefinite optimization oracle that solves a convex feasibility problem at hand in every iteration for different values of Q. In particular, these facts, along with Lemmas 14 and 15 ensure that the complexity results in Propositions 6 and 7 hold when applying Algorithm 1 to solve (14).

Proposition 8. *Let $Q \circ \widetilde{C} \in \mathcal{S}^n$ be stored in QRAM. Algorithm 1 solves* (14) *up to additive error ϵ using*

$$\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}\left(n^{1.5}\epsilon^{-5}\right)$$

accesses to the QRAM.

Proof. Given that $Q \circ \widetilde{C}$ is stored in QRAM, Lemma 3*(ii)* asserts that when constructing a block-encoding of $Q \circ \widetilde{C}$, one can set the subnormalization factor to be $\alpha_C = \left\|Q \circ \widetilde{C}\right\|_F$. In particular, one can always choose $\alpha_C = 1$, as it can be seen from the proof of Lemma 15 that the inequality

$$\left\|Q \circ \widetilde{C}\right\|_F \leq \left\|\widetilde{C}\right\|_F = 1$$

always holds for any Q defined according to equation (13). Collecting these facts, one can construct a $(1, \mathcal{O}(\log(n)), \epsilon/(2n))$-block-encoding of $Q \circ \widetilde{C}$ in time $\widetilde{\mathcal{O}}_{\frac{n}{\epsilon}}(1)$. Note that the quantity $Q \circ \widetilde{C}$ remains unchanged for the duration of Algorithm 1. From here, the rest of the proof follows exactly that of Proposition 7 upon replacing $\widetilde{C}$, $O_{\mathcal{C}_\gamma}$ and $O_{\mathcal{D}_n}$ with $Q \circ \widetilde{C}$, $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$, respectively, in what remains. □

Before proceeding further, we establish that the eigenvalues of the updated solution will never fall significantly below zero by deriving a lower bound on the minimum eigenvalue of the terms $\frac{1}{\eta}Q \circ \rho$ that are used to update the overall solution in each iteration of our refinement scheme according to (12).

Proposition 9. *Let ρ be a solution to* (14) *obtained from running Algorithm 1 using precision $\epsilon \in (0, 1)$. Then,*

$$\frac{1}{\eta}Q \circ \rho \succeq -2 \cdot \left(\|\varepsilon\|_1 + \frac{\epsilon}{\eta}\right) n^{-1} I.$$

Proof. In what follows, we assume without loss of generality that Q has at least one negative eigenvalue (otherwise, $Q \circ \rho \succeq 0$ trivially holds), so applying Lemma 13 we can let $\lambda_{\min}(Q) \geq -2$. Applying Lemma 2, we can lower bound the minimum eigenvalue of the Hadamard product $Q \circ \rho$ as follows

$$\lambda_{\min}(Q \circ \rho) \geq \min_{i \in [n]} \rho_{ii} \cdot \lambda_{\min}(Q) \geq -2 \min_{i \in [n]} \rho_{ii}.$$

Therefore, in order to derive a worst case lower bound on $\lambda_{\min}(Q \circ \rho)$, it suffices to determine

$$\max_{\rho \in \mathcal{D}_{\eta\varepsilon}} \min_{i \in [n]} \rho_{ii}.$$

The definition of $\mathcal{D}_{\eta\varepsilon}$ asserts that when $O_{\mathcal{D}_{\eta\varepsilon}}$ is queried with precision ϵ, the diagonal elements of ρ are nonnegative and must satisfy the following:

$$\sum_{i \in [n]} |\rho_{ii} - \eta|\varepsilon_i|| \leq \epsilon, \quad \sum_{i \in [n]} \rho_{ii} \leq \eta\|\varepsilon\|_1 + \epsilon.$$

Hence, $\max_{\rho \in \mathcal{D}_{\eta\varepsilon}} \min_{i \in [n]} \rho_{ii} \leq \frac{\eta\|\varepsilon\|_1 + \epsilon}{n}$, and the proof is complete. □

4.3 Iterative Refinement using Hamiltonian Updates

We are now in a position to provide our iterative refinement method for SDO approximations of QUBOs presented in full detail in Algorithm 2.

The algorithm takes three parameters as input; *(i)* ξ, the fixed (constant) precision used to test closeness to the sets $\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}$ and $\mathcal{D}_{\eta\varepsilon}$ in every iteration, *(ii)* ζ, the precision to which the final solves (5), and *(iii)* ϵ, the additive error to which we seek to solve (3). In our initialization steps we set the values of Q, ε and η such that the call to Algorithm 1 corresponds to solving the original feasibility problem (10).

Algorithm 2 Iterative Refinement for SDO Approximations of QUBOs

Input: Error tolerances $\epsilon \in (0,1)$ and $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$, upper bound on objective value $\gamma \in [-1,1]$

Output: A matrix $\tilde{\rho} \in \left\{X \in \mathcal{S}_+^n : \mathrm{tr}(X) \leq 1+\zeta\right\}$ satisfying

$$\max\left\{\gamma - \mathrm{tr}\left(\widetilde{C}\tilde{\rho}\right), \left\|\sum_{i\in[n]} \langle i|\,\tilde{\rho}\,|i\rangle |i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}}\right\} \leq \zeta$$

Initialize: $\tilde{\rho}, \hat{\rho} \leftarrow \mathbf{0}^{n\times n}$, $Q \leftarrow ee^\top$, $\varepsilon_i = \frac{1}{n}$ for $i \in [n]$, $\tilde{\gamma}, \hat{\gamma} \leftarrow 0$, $\eta^{(0)} \leftarrow 1$, $k \leftarrow 1$

$\tilde{\rho}^{(0)} \leftarrow$ solve (14) to precision $\frac{\xi^2}{16}$ using Algorithm 1 with oracles $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$

$\tilde{\gamma}^{(0)} \leftarrow \mathrm{tr}\left(\widetilde{C}\tilde{\rho}^{(0)}\right)$

$\varepsilon_i^{(0)} \leftarrow \tilde{\rho}_{ii}^{(0)} - \frac{1}{n}$ for $i \in [n]$

$Q_{ii} \leftarrow \mathrm{sign}(-\varepsilon_i^{(0)})$ for $i \in [n]$

$\eta^{(1)} \leftarrow \frac{1}{\max\left\{\gamma - \tilde{\gamma}^{(0)}, \|\varepsilon^{(0)}\|_1\right\}}$

$\delta^{(1)} \leftarrow \frac{2}{n}\left(\|\varepsilon^{(0)}\|_1 + \frac{(\xi/4)^2}{\eta^{(1)}}\right)$

while $\max\left\{\gamma - \mathrm{tr}\left(\widetilde{C}\tilde{\rho}\right), \|\varepsilon\|_1\right\} > \zeta$ **do**

1. Store refining problem data $\left(Q \circ \widetilde{C}, \eta^{(k)}\varepsilon^{(k-1)}, \eta^{(k)}\tilde{\gamma}^{(k-1)}\right)$
2. Solve (14) to precision $\frac{\xi^2}{16}$ for $\rho^{(k)}$ using Algorithm 1 with oracles $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$
3. Update solution
$$\hat{\rho}^{(k)} \leftarrow \tilde{\rho}^{(k-1)} + \frac{1}{\eta^{(k)}} Q \circ \rho^{(k)}$$
4. Apply spectrum shift to $\hat{\rho}^{(k)}$ to obtain a positive semidefinite matrix
$$\tilde{\rho}^{(k)} \leftarrow \frac{1}{1+n\delta^{(k)}}\left(\hat{\rho}^{(k)} + \delta^{(k)} I\right)$$
5. Update objective value and compute element-wise deviations from the maximally mixed state:
$$\tilde{\gamma}^{(k)} \leftarrow \mathrm{tr}\left(\widetilde{C}\tilde{\rho}^{(k)}\right), \quad \varepsilon_i^{(k)} \leftarrow \tilde{\rho}_{ii}^{(k)} - \frac{1}{n} \text{ for } i \in [n]$$
6. Update refining problem parameters:
$$Q_{ii} \leftarrow \mathrm{sign}\left(-\varepsilon_i^{(k)}\right) \text{ for } i \in [n], \quad \eta^{(k+1)} \leftarrow \frac{1}{\max\left\{\gamma - \tilde{\gamma}^{(k)}, \left\|\varepsilon^{(k)}\right\|_1\right\}}$$
7. Update spectrum shift parameter:
$$\delta^{(k+1)} \leftarrow \frac{2}{n}\left(\left\|\varepsilon^{(k)}\right\|_1 + \frac{(\xi/4)^2}{\eta^{(k+1)}}\right)$$
8. $k \leftarrow k+1$

end

In each iteration k, Algorithm 2 calls Algorithm 1 with separation oracles $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$ using fixed precision ξ^2 such that every call to Algorithm 1 produces a ξ^2-precise solution $\rho^{(k)}$ to (14). Using $\rho^{(k)}$ to update our solution according to (12) may cause us to obtain matrices $\hat{\rho}^{(k)}$ with eigenvalues that are, in the worst case, slightly negative. A shift of the spectrum defined via the bound on the minimum eigenvalue of the update term $\frac{1}{\eta^{(k)}} Q \circ \rho^{(k)}$ provided in Proposition 9 suffices to obtain a positive semidefinite matrix $\tilde{\rho}^{(k)}$, and we will demonstrate that it does not change the constraint violation or the objective function value by a large amount.

If $\tilde{\rho}$ is indistinguishable up to precision ζ from the maximally mixed state $n^{-1}I$ upon measurement in the computational basis, and satisfies $\operatorname{tr}\left(\widetilde{C}\tilde{\rho}\right) \geq \gamma - \zeta$, the algorithm terminates and reports $\tilde{\rho}$. Otherwise, we shift the spectrum as described above, we construct the refining problem associated with our current solution, and proceed to the next iteration. To define the parameters for the next refining problem, we first calculate the deviation of the diagonal elements from $\frac{1}{n}$, and the violation with respect to satisfying our objective value. Then, we define our scaling factor to be the reciprocal of the maximum over the ℓ_1-norm of the diagonal deviations, and the objective violation. We stress that ξ is a (chosen) constant, and does not change throughout the algorithm.

We now state a series of results in order to bound the iteration complexity of Algorithm 2, and use our findings to improve the approximation guarantee given in Proposition 5. We begin by proving establishing that the iterates generated by Algorithm 2 are increasingly accurate solutions to (5).

Theorem 4. *Fix a constant $\xi \in \left(0, \frac{1}{2}\right)$ and define $\tilde{\xi} = \frac{\xi}{4}$. Let $\rho^{(k)}$ be a solution to (14) obtained from running Algorithm 1 using fixed precision $\tilde{\xi}^2$ in iteration k of Algorithm 2. Then, the following hold:*

(a) For $k \geq 0$, $\eta^{(k)} \geq \frac{1}{2\xi^k}$.

(b) For $k \geq 1$, $\hat{\rho}^{(k)} = \tilde{\rho}^{(k-1)} + \frac{1}{\eta^{(k)}} Q \circ \rho^{(k)}$ satisfies

$$\max\left\{\gamma - \operatorname{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right), \left\|\sum_{i\in[n]} \langle i|\,\hat{\rho}^{(k)}\,|i\rangle |i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}}\right\} \leq \xi^{k+2}, \quad \lambda_{\min}\left(\hat{\rho}^{(k)}\right) \geq -\frac{\xi^{k+1}}{n}.$$

(c) For $k \geq 0$, $\tilde{\rho}^{(k)}$ satisfies

$$\max\left\{\gamma - \operatorname{tr}\left(\widetilde{C}\tilde{\rho}^{(k)}\right), \left\|\sum_{i\in[n]} \langle i|\,\tilde{\rho}^{(k)}\,|i\rangle |i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}}\right\} \leq 2\xi^{k+1}, \quad \lambda_{\min}\left(\tilde{\rho}^{(k)}\right) \geq 0.$$

That is, $\tilde{\rho}^{(k)}$ is an $2\xi^{k+1}$-precise solution to (5).

Proof. First, observe that that we initialize $\varepsilon_i = \frac{1}{n}$ for $i \in [n]$, $\tilde{\gamma} = 0$, $\eta^{(0)} = 1$ and $Q = ee^\top$. Under these conditions, one can observe that if $\tilde{\rho}^{(0)}$ is obtained from solving (14) to precision $\tilde{\xi}^2$ using the oracles $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$, we must have

$$\sum_{i=1}^{n} \left|\left\langle i \left| \tilde{\rho}^{(0)} \right| i \right\rangle - \frac{1}{n}\right| \leq \frac{\tilde{\xi}^2}{\eta^{(0)}} = \tilde{\xi}^2. \tag{15}$$

In other words, $\tilde{\rho}^{(0)}$ satisfies

$$\left\|\sum_{i\in[n]} \langle i|\,\tilde{\rho}^{(0)}\,|i\rangle |i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}} \leq \tilde{\xi}^2,$$

and by the definition of $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ we also have

$$\operatorname{tr}\left(\widetilde{C}\tilde{\rho}^{(0)}\right) \geq \gamma - \frac{\tilde{\xi}^2}{\eta^{(0)}} = \gamma - \tilde{\xi}^2. \tag{16}$$

Since $\tilde{\rho}^{(0)} \succeq 0$ by construction, clearly $\tilde{\rho}^{(0)}$ is a ξ^2-precise solution to (5).

Next, we proceed by induction to establish that for $k \geq 1$, the matrix $\hat{\rho}^{(k)}$ satisfies

$$\max\left\{\gamma - \operatorname{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right), \left\|\sum_{i\in[n]} \langle i|\,\hat{\rho}^{(k)}\,|i\rangle |i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}}\right\} \leq \frac{\tilde{\xi}^2}{\eta^{(k)}},\ \lambda_{\min}\left(\hat{\rho}^{(k)}\right) \geq -\frac{2}{n}\left(\frac{\tilde{\xi}^2}{\eta^{(k-1)}} + \frac{\tilde{\xi}^2}{\eta^{(k)}}\right). \quad (17)$$

For all $k \geq 1$, we have $\varepsilon_i^{(k-1)} = \tilde{\rho}_{ii}^{(k-1)} - \frac{1}{n}$ for $i \in [n]$ and $Q = (ee^\top - I) + \operatorname{diag}\left(\operatorname{sign}\left(-\varepsilon^{(k-1)}\right)\right)$. For this choice of parameters, the general feasibility problem (14) reduces to the refining problem (11) and the solution $\rho^{(k)}$ obtained via Algorithm 1 using the oracles $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$ must satisfy

$$\operatorname{tr}\left(\widetilde{C}Q \circ \rho^{(k)}\right) \geq \eta^{(k)}\left(\gamma - \tilde{\gamma}^{(k-1)}\right) - \tilde{\xi}^2 \quad (18a)$$

$$\sum_{i=1}^{n}\left|\left\langle i\left|\rho^{(k)}\right|i\right\rangle - \eta^{(k)}|\varepsilon_i|\right| \leq \tilde{\xi}^2. \quad (18b)$$

Accordingly, for $k \geq 1$, setting $\hat{\rho}^{(k)} = \tilde{\rho}^{(k-1)} + \frac{1}{\eta^{(k)}}Q \circ \rho^{(k)}$ we can bound the total infeasibility of the diagonal constraints as follows

$$\begin{aligned}
\sum_{i=1}^{n}\left|\left\langle i\left|\hat{\rho}^{(k)}\right|i\right\rangle - \frac{1}{n}\right| &= \sum_{i=1}^{n}\left|\left\langle i\left|\tilde{\rho}^{(k-1)} + \frac{1}{\eta^{(k)}}Q \circ \rho^{(k)}\right|i\right\rangle - \frac{1}{n}\right| \\
&= \sum_{i=1}^{n}\left|\left(\tilde{\rho}_{ii}^{(k-1)} + \frac{1}{\eta^{(k)}}\left(\operatorname{sign}\left(-\varepsilon_i^{(k-1)}\right)\cdot\rho_{ii}^{(k)}\right)\right) - \frac{1}{n}\right| \\
&= \sum_{i=1}^{n}\left|\left(\tilde{\rho}_{ii}^{(k-1)} - \frac{1}{n}\right) + \frac{1}{\eta^{(k)}}\operatorname{sign}\left(-\varepsilon_i^{(k-1)}\right)\rho_{ii}^{(k)}\right| \\
&= \sum_{i=1}^{n}\left|\varepsilon_i^{(k-1)} + \frac{1}{\eta^{(k)}}\operatorname{sign}\left(-\varepsilon_i^{(k-1)}\right)\rho_{ii}^{(k)}\right| \\
&= \frac{1}{\eta^{(k)}}\sum_{i=1}^{n}\left|\eta^{(k)}\varepsilon_i^{(k-1)} + \operatorname{sign}\left(-\varepsilon_i^{(k-1)}\right)\rho_{ii}^{(k)}\right| \leq \frac{\tilde{\xi}^2}{\eta^{(k)}},
\end{aligned}$$

where the final inequality follows from (18b). Consequently, we can conclude that at iteration $k \geq 1$ we have

$$\left\|\sum_{i\in[n]} \langle i|\,\hat{\rho}^{(k)}\,|i\rangle |i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}} \leq \frac{\tilde{\xi}^2}{\eta^{(k)}}. \quad (19)$$

Next, letting $\tilde{c}^{(k)} = \operatorname{tr}\left(\widetilde{C}Q \circ \rho^{(k)}\right)$, one can observe

$$\operatorname{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right) = \operatorname{tr}\left(\widetilde{C}\left(\tilde{\rho}^{(k-1)} + \frac{1}{\eta^{(k)}}Q \circ \rho^{(k)}\right)\right) = \operatorname{tr}\left(\widetilde{C}\tilde{\rho}^{(k-1)}\right) + \frac{1}{\eta^{(k)}}\operatorname{tr}\left(\widetilde{C}Q \circ \rho^{(k)}\right) = \tilde{\gamma}^{(k-1)} + \frac{\tilde{c}^{(k)}}{\eta^{(k)}}.$$

Since $\tilde{c}^{(k)} \geq \eta^{(k)}\left(\gamma - \tilde{\gamma}^{(k-1)}\right) - \xi^2$ due to (18a), it follows:

$$\operatorname{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right) = \tilde{\gamma}^{(k-1)} + \frac{\tilde{c}^{(k)}}{\eta^{(k)}} \geq \tilde{\gamma}^{(k-1)} + \frac{1}{\eta^{(k)}}\left[\eta^{(k)}\left(\gamma - \tilde{\gamma}^{(k-1)}\right) - \tilde{\xi}^2\right] = \gamma - \frac{\tilde{\xi}^2}{\eta^{(k)}}. \quad (20)$$

To prove the eigenvalue bound on $\hat{\rho}^{(k)}$, we first establish that $\tilde{\rho}^{(k)} \succeq 0$ holds for all $k \geq 0$ using induction. When $k = 0$, this is trivially true because $\tilde{\rho}^{(0)} \succ 0$ by construction (it is a Gibbs state). Now assume that $\tilde{\rho}^{(\ell)} \succeq 0$ holds for all $\ell = 1, \ldots, k-1$. At the k-th iterate, we have

$$\tilde{\rho}^{(k)} := \frac{1}{1 + n\delta^{(k)}}\left(\hat{\rho}^{(k)} + \delta^{(k)}I\right) = \frac{1}{1 + n\delta^{(k)}}\left[\tilde{\rho}^{(k-1)} + \frac{1}{\eta^{(k)}}Q \circ \rho^{(k)} + \delta^{(k)}I\right].$$

Since we define $\delta^{(k)} = \frac{2}{n}\left(\left\|\varepsilon^{(k-1)}\right\|_1 + \frac{\tilde{\xi}^2}{\eta^{(k)}}\right)$, Proposition 9 asserts that $\frac{1}{\eta^{(k)}} Q \circ \rho^{(k)} + \delta^{(k)} I \succeq 0$. Combining this fact with $\tilde{\rho}^{(k-1)} \succeq 0$, we have $\tilde{\rho}^{(k)} \succeq 0$ because it is defined as the sum of two symmetric positive semidefinite matrices, thus completing the induction argument.

Having shown $\tilde{\rho}^{(k)} \succeq 0$ holds for all $k \geq 0$, it follows that $\hat{\rho}^{(k)}$ is defined as the sum of a positive semidefinite matrix, and a symmetric matrix satisfying $\frac{1}{\eta^{(k)}} Q \circ \rho^{(k)} \succeq -\delta^{(k)} I$. To see this, first observe that the residuals along the diagonal ε are always computed with respect to $\tilde{\rho}$, and in particular for $k \geq 1$

$$\begin{aligned}
\left\|\varepsilon^{(k)}\right\|_1 = \left\|\sum_{i\in[n]} \langle i|\, \tilde{\rho}^{(k)}\, |i\rangle |i\rangle\langle i| - n^{-1} I\right\|_{\text{tr}} &= \sum_{i\in[n]} \left|\frac{1}{1+n\delta^{(k)}}\left(\hat{\rho}_{ii}^{(k)} + \delta^{(k)}\right) - n^{-1}\right| \\
&= \frac{1}{1+n\delta^{(k)}} \sum_{i\in[n]} \left|\hat{\rho}_{ii}^{(k)} + \delta^{(k)} - n^{-1}\left(1+n\delta^{(k)}\right)\right| \\
&= \frac{1}{1+n\delta^{(k)}} \sum_{i\in[n]} \left|\hat{\rho}_{ii}^{(k)} + \delta^{(k)} - n^{-1} - \delta^{(k)}\right| \\
&\leq \frac{1}{1+n\delta^{(k)}} \sum_{i\in[n]} \left|\hat{\rho}_{ii}^{(k)} - n^{-1}\right| \\
&\leq \frac{1}{1+n\delta^{(k)}} \cdot \frac{\tilde{\xi}^2}{\eta^{(k)}}, \qquad (21)
\end{aligned}$$

where the final inequality follows from (19). Thus, applying Lemma 12 along with the definition of $\delta^{(k)}$ and noting (15), (21) we obtain

$$\delta^{(k)} = \frac{2}{n}\left(\left\|\varepsilon^{(k-1)}\right\|_1 + \frac{\tilde{\xi}^2}{\eta^{(k)}}\right) \leq \frac{2}{n}\left(\frac{\tilde{\xi}^2}{\eta^{(k-1)}} + \frac{\tilde{\xi}^2}{\eta^{(k)}}\right) \implies \hat{\rho}^{(k)} \succeq -2\left(\frac{\tilde{\xi}^2}{\eta^{(k-1)}} + \frac{\tilde{\xi}^2}{\eta^{(k)}}\right) n^{-1} I, \qquad (22)$$

for all $k \geq 1$. Hence, from (19), (20) and (22), the matrix $\hat{\rho}^{(k)} = \tilde{\rho}^{(k-1)} + \frac{1}{\eta^{(k)}} Q \circ \rho^{(k)}$ satisfies (17) for all $k \geq 0$.

Next, note that the spectrum shift used to restore positive semidefiniteness is mild. Indeed for all $k \geq 1$

$$\begin{aligned}
\left\|\hat{\rho}^{(k)} - \tilde{\rho}^{(k)}\right\|_{\text{tr}} = \left\|\hat{\rho}^{(k)} - \frac{1}{1+n\delta^{(k)}}\left(\hat{\rho}^{(k)} + \delta^{(k)} I\right)\right\|_{\text{tr}} &= \left\|\frac{1+n\delta^{(k)}-1}{1+n\delta^{(k)}}\hat{\rho}^{(k)} - \frac{1}{1+n\delta^{(k)}}\delta^{(k)} I\right\|_{\text{tr}} \\
&\leq \frac{n\delta^{(k)}}{1+n\delta^{(k)}}\left\|\hat{\rho}^{(k)} - n^{-1} I\right\|_{\text{tr}} \\
&= \frac{n\delta^{(k)}}{1+n\delta^{(k)}}\left\|\hat{\rho}^{(k)} - n^{-1} I + (\delta^{(k)} I - \delta^{(k)} I)\right\|_{\text{tr}} \\
&\leq \frac{n\delta^{(k)}}{1+n\delta^{(k)}}\left[\left\|\hat{\rho}^{(k)} + \delta^{(k)} I\right\|_{\text{tr}} + \left\|n^{-1} I\right\|_{\text{tr}} + \left\|\delta^{(k)} I\right\|_{\text{tr}}\right] \\
&\leq \frac{n\delta^{(k)}}{1+n\delta^{(k)}}\left[1 + \frac{\tilde{\xi}^2}{\eta^{(k)}} + n\delta^{(k)} + 1 + n\delta^{(k)}\right] \\
&= \left(2 + \frac{\tilde{\xi}^2}{\eta^{(k)}} + 2n\delta^{(k)}\right)\frac{n\delta^{(k)}}{1+n\delta^{(k)}},
\end{aligned}$$

where the second to last inequality follows from the fact that $\hat{\rho}^{(k)} + \delta^{(k)} I$, $n^{-1} I$ and $\delta^{(k)} I$ are all positive semidefinite matrices.

Using the definition of $\delta^{(k)}$, we can continue the chain of inequalities:

$$\begin{aligned}\left(2+\frac{\tilde{\xi}^2}{\eta^{(k)}}+2n\delta^{(k)}\right)\frac{n\delta^{(k)}}{1+n\delta^{(k)}} &< \left(2+\frac{\tilde{\xi}^2}{\eta^{(k)}}+2n\delta^{(k)}\right)n\delta^{(k)} \\ &\leq 2\left(2+3\frac{\tilde{\xi}^2}{\eta^{(k)}}+2\left\|\varepsilon^{(k-1)}\right\|_1\right)\left(\left\|\varepsilon^{(k-1)}\right\|_1+\frac{\tilde{\xi}^2}{\eta^{(k)}}\right) \\ &= 4\left[\left\|\varepsilon^{(k-1)}\right\|_1+\frac{5}{2}\frac{\tilde{\xi}^2}{\eta^{(k)}}\left\|\varepsilon^{(k-1)}\right\|_1+\frac{3}{2}\frac{\tilde{\xi}^4}{\left(\eta^{(k)}\right)^2}+\left\|\varepsilon^{(k-1)}\right\|_1^2\right] \\ &\leq 4\left[\frac{\tilde{\xi}^2}{\eta^{(k-1)}}+\frac{5}{2}\frac{\tilde{\xi}^2}{\eta^{(k-1)}}\frac{\tilde{\xi}^2}{\eta^{(k)}}+\frac{3}{2}\frac{\tilde{\xi}^4}{\left(\eta^{(k)}\right)^2}+\frac{\tilde{\xi}^4}{\left(\eta^{(k-1)}\right)^2}\right],\end{aligned}$$

where the final inequality follows upon noting (15) and (21). For ease of notation, for all $k \geq 1$ define

$$\Phi\left(\eta^{(k-1)},\eta^{(k)},\tilde{\xi}\right) := 4\left[\frac{\tilde{\xi}^2}{\eta^{(k-1)}}+\frac{5}{2}\frac{\tilde{\xi}^2}{\eta^{(k-1)}}\frac{\tilde{\xi}^2}{\eta^{(k)}}+\frac{3}{2}\frac{\tilde{\xi}^4}{\left(\eta^{(k)}\right)^2}+\frac{\tilde{\xi}^4}{\left(\eta^{(k-1)}\right)^2}\right].$$

Applying a matrix Hölder inequality, one can observe that for all $k \geq 1$:

$$\left|\mathrm{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right)-\mathrm{tr}\left(\widetilde{C}\tilde{\rho}^{(k)}\right)\right| \leq \left\|\widetilde{C}\right\|\left\|\hat{\rho}^{(k)}-\tilde{\rho}^{(k)}\right\|_{\mathrm{tr}} \leq \left\|\hat{\rho}^{(k)}-\tilde{\rho}^{(k)}\right\|_{\mathrm{tr}} < \Phi\left(\eta^{(k-1)},\eta^{(k)},\tilde{\xi}\right),$$

from which we can conclude

$$\begin{aligned}\gamma-\mathrm{tr}\left(\widetilde{C}\tilde{\rho}^{(k)}\right) &= \gamma-\mathrm{tr}\left(\widetilde{C}\tilde{\rho}^{(k)}\right)+\left[\mathrm{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right)-\mathrm{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right)\right] \\ &= \gamma-\mathrm{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right)+\left[\mathrm{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right)-\mathrm{tr}\left(\widetilde{C}\tilde{\rho}^{(k)}\right)\right] \leq \frac{\tilde{\xi}^2}{\eta^{(k)}}+\Phi\left(\eta^{(k-1)},\eta^{(k)},\tilde{\xi}\right), \qquad (23)\end{aligned}$$

holds for all $k \geq 1$, as $\gamma-\mathrm{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right) \leq \frac{\tilde{\xi}^2}{\eta^{(k)}}$. We can now use this fact to establish the lower bound on $\eta^{(k)}$, which we prove by induction.

For $k=0$, we have $\eta^{(0)}=1$ and $\eta^{(1)}=\frac{1}{\xi^2}$ when $k=1$ due to (15) and (16). for which $\eta^{(k)} \geq \frac{1}{2\cdot\xi^k}$ trivially holds. By the induction hypothesis, it assumed that $\eta^{(\ell)} \geq \frac{1}{2\xi^\ell}$ is true for $\ell=2,\ldots,k$. From here, one can observe that

$$\begin{aligned}\Phi\left(\eta^{(k-1)},\eta^{(k)},\tilde{\xi}\right) &= 4\left[\frac{\tilde{\xi}^2}{\eta^{(k-1)}}+\frac{5}{2}\frac{\tilde{\xi}^2}{\eta^{(k-1)}}\frac{\tilde{\xi}^2}{\eta^{(k)}}+\frac{3}{2}\frac{\tilde{\xi}^4}{\left(\eta^{(k)}\right)^2}+\frac{\tilde{\xi}^4}{\left(\eta^{(k-1)}\right)^2}\right] \\ &\leq 4\left[2\tilde{\xi}^2\xi^{k-1}+\frac{5}{2}\tilde{\xi}^4\xi^{k-1}\xi^k+6\tilde{\xi}^4\xi^{2k}+4\tilde{\xi}^4\xi^{2[k-1]}\right] \\ &= 8\tilde{\xi}^2\xi^{k-1}+10\tilde{\xi}^4\xi^{k-1}\xi^k+24\tilde{\xi}^4\xi^{2k}+16\tilde{\xi}^4\xi^{2[k-1]} \\ &\leq \frac{8}{16}\xi^{k+1}+\frac{10}{256}\xi^{2k+3}+\frac{24}{256}\xi^{2k+4}+\frac{16}{256}\xi^{2k+2} \\ &< \frac{178}{256}\xi^{k+1}.\end{aligned}$$

Noting $\Phi\left(\eta^{(k-1)}, \eta^{(k)}, \tilde{\xi}\right) < \frac{86}{64}\xi^{k+1}$ and applying (17) yields

$$\begin{aligned}\eta^{(k+1)} = \frac{1}{\max\left\{\gamma - \operatorname{tr}\left(\widetilde{C}\tilde{\rho}^{(k)}\right), \left\|\sum_{i\in[n]} \langle i|\,\tilde{\rho}^{(k)}\,|i\rangle|i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}}\right\}} &\geq \frac{1}{\frac{\tilde{\xi}^2}{\eta^{(k)}} + \Phi\left(\eta^{(k-1)}, \eta^{(k)}, \tilde{\xi}\right)} \\ &\geq \frac{1}{\frac{\tilde{\xi}^2}{\eta^{(k)}} + \frac{178}{256}\xi^{k+1}} \\ &\geq \frac{1}{2\tilde{\xi}^2\xi^k + \frac{178}{256}\xi^{k+1}} > \frac{1}{2\xi^{k+1}},\end{aligned}$$

which completes the proof of *(a)*.

Having demonstrated that *(a)* holds, to prove *(b)*, we can simply combine inequality (17) with the lower bound $\eta^{(k)} \geq \frac{1}{2\xi^k}$, which together imply

$$\max\left\{\gamma - \operatorname{tr}\left(\widetilde{C}\hat{\rho}^{(k)}\right), \left\|\sum_{i\in[n]} \langle i|\,\hat{\rho}^{(k)}\,|i\rangle|i\rangle\langle i| - n^{-1}I\right\|_{\mathrm{tr}}\right\} \leq \frac{\tilde{\xi}^2}{\eta^{(k)}} \leq \xi^{k+2}, \quad \lambda_{\min}\left(\hat{\rho}^{(k)}\right) \geq -\frac{\xi^{k+1}}{n}.$$

Upon noting (21), (23) and that $\tilde{\rho}^{(k)} \succeq 0$ always holds, the result in *(c)* follows from a similar argument. □

The next result establishes polynomial convergence of Algorithm 2.

Corollary 4. *Let $0 < \zeta \ll \xi < 1$, and $\eta^{(0)} = 1$. Then, Algorithm 2 terminates in at most*

$$K = \mathcal{O}\left(\log\left(\frac{1}{\zeta}\right)\right)$$

iterations.

Proof. The result follows from Theorem 4*(c)*. □

It is important at this point for us to remark that fixing $\xi \in (0,1)$ does not limit us with respect to how accurately we can solve (3). We can always make the final precision parameter arbitrarily small using only $\widetilde{\mathcal{O}}_{\frac{1}{\xi}}(1)$ iterations, as the overall running time depends only poly-logarithmically on ζ^{-1}. Accordingly, we take advantage of this fact and revisit the approximation guarantee provided in Proposition 5.

Proposition 10. *Let $\tilde{\rho}$ be a ζ-accurate solution to the renormalized and relaxed SDO problem* (5) *with input matrix C and $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$. Let $\gamma_\zeta = \operatorname{tr}(C\tilde{\rho})$ be the value attained by $\tilde{\rho}$. Then, there is a quantum state ρ^* at trace distance $\mathcal{O}\left(\frac{\epsilon}{n\|C\|_F}\right)$ of $\tilde{\rho}$ such that $n\rho^*$ is a feasible point of SDO problem* (3)*. In particular*

$$|\gamma_\zeta n\|C\|_F - \operatorname{tr}(n\rho^* C)| = \mathcal{O}(\epsilon).$$

Moreover, it is possible to construct ρ^* in time $\mathcal{O}(n^2)$ given the entries of $\tilde{\rho}$.

Proof. The proof almost exactly follows the proof of Proposition 3.1 in [11], regardless, we present the adjusted proof for completeness. Our aim is to show that a ζ-precise solution $\tilde{\rho}$ to (5) obtained using Algorithm 2 can be used to construct ρ^* such that $n\rho^*$ is an exactly feasible solution to (3).

We begin by examining the diagonal elements of $\tilde{\rho}$ and check whether modifications need to be made to ensure that our solution is an exactly feasible point to the renormalized SDO problem (5). Namely, if $|\langle i|\tilde{\rho}|i\rangle - \frac{1}{n}| > \frac{\sqrt{\zeta}}{n}$ for $i \in [n]$, we replace $\tilde{\rho}_{ii}$ with $\frac{1}{n}$ and set all elements in the i-th row and the i-th column to 0, and denote the resulting matrix by ρ'. From here we introduce another matrix W which we obtain

by replacing each diagonal entry of ρ' with $\frac{1}{n}$. In general we may not have $W \succeq 0$, so the authors in [11] suggest using the convex combination:

$$\rho^* = \frac{1}{1+\sqrt{\zeta}}\left(W + \frac{\sqrt{\zeta}}{n} I\right).$$

Then, $\rho^* \succeq 0$ and by construction $\langle i|\rho^*|i\rangle = \frac{1}{n}$ for all $i \in [n]$. Hence, ρ^* is a feasible solution to the renormalized SDO problem (5).

What remains is to show that the above reformulations yield the desired approximation. Denote by $\mathcal{B} = \{i : |n\langle i|\tilde{\rho}|i\rangle - 1| > \sqrt{\zeta}\} \subset [n]$ the set of diagonal entries that deviate substantially from $\frac{1}{n}$. Without loss of generality, it suffices to assume that such elements are found in the first $|\mathcal{B}|$ rows of $\tilde{\rho}$, in which case

$$\begin{aligned}\|\rho' - \tilde{\rho}\|_{\text{tr}} = \left\| \begin{pmatrix} n^{-1} I_{\mathcal{B}} & 0 \\ 0 & \tilde{\rho}_{22} \end{pmatrix} - \begin{pmatrix} \tilde{\rho}_{11} & \tilde{\rho}_{12} \\ \tilde{\rho}_{21} & \tilde{\rho}_{22} \end{pmatrix} \right\|_{\text{tr}} &= \left\| \begin{pmatrix} n^{-1} I_{\mathcal{B}} - \tilde{\rho}_{11} & -\tilde{\rho}_{12} \\ -\tilde{\rho}_{21} & 0 \end{pmatrix} \right\|_{\text{tr}} \\ &\leq \|\tilde{\rho}_{11}\|_{\text{tr}} + 2\|\tilde{\rho}_{12}\|_{\text{tr}} + \|n^{-1} I_{\mathcal{B}}\|_{\text{tr}}. \end{aligned} \tag{24}$$

Since $\tilde{\rho}$ is a ζ-precise solution to (5), $\tilde{\rho}$ obeys

$$\sum_{i=1}^{n} \left| \langle i|\tilde{\rho}|i\rangle - \frac{1}{n} \right| \leq \zeta.$$

Therefore, we must have

$$|\mathcal{B}| \frac{\sqrt{\zeta}}{n} \leq \zeta,$$

which equates to $|\mathcal{B}| \leq n\sqrt{\zeta}$. Now, by the definition of $\mathcal{B}$, it follows

$$\|\tilde{\rho}_{22}\|_{\text{tr}} \geq (n - |\mathcal{B}|)\frac{1-\sqrt{\zeta}}{n} \geq (n - n\sqrt{\zeta})\frac{1-\sqrt{\zeta}}{n} = (1-\sqrt{\zeta})^2.$$

Following [11], we invoke a result from [41], which states

$$\left\| \begin{bmatrix} \|\tilde{\rho}_{11}\|_{\text{tr}} & \|\tilde{\rho}_{12}\|_{\text{tr}} \\ \|\tilde{\rho}_{12}^\top\|_{\text{tr}} & \|\tilde{\rho}_{22}\|_{\text{tr}} \end{bmatrix} \right\|_{\text{tr}} \leq \left\| \begin{bmatrix} \tilde{\rho}_{11} & \tilde{\rho}_{12} \\ \tilde{\rho}_{12}^\top & \tilde{\rho}_{22} \end{bmatrix} \right\|_{\text{tr}} = \|\tilde{\rho}\|_{\text{tr}} = \text{tr}\,(\tilde{\rho}) = 1.$$

Using the fact that $\|\cdot\|_{\text{tr}} \geq \|\cdot\|_2$, where $\|\cdot\|_2$ is the Frobenius, or Schatten-2 norm, the above implies

$$\|\tilde{\rho}_{11}\|_{\text{tr}}^2 + 2\|\tilde{\rho}_{12}\|_{\text{tr}}^2 + \|\tilde{\rho}_{22}\|_{\text{tr}}^2 \leq 1.$$

As $\|\tilde{\rho}_{22}\|_{\text{tr}} \geq (1-\sqrt{\zeta})^2$, it can be seen trivially that $\|\tilde{\rho}_{22}\|_{\text{tr}}^2 \geq (1-\sqrt{\zeta})^4$, and thus

$$\|\tilde{\rho}_{11}\|_{\text{tr}}^2 + 2\|\tilde{\rho}_{12}\|_{\text{tr}}^2 \leq 1 - (1-\sqrt{\zeta})^4 = \mathcal{O}(\sqrt{\zeta}).$$

Consequently $\|\tilde{\rho}_{11}\|_{\text{tr}} + 2\|\tilde{\rho}_{12}\|_{\text{tr}} = \mathcal{O}\left(\zeta^{\frac{1}{4}}\right)$, and plugging this into equation (24) asserts

$$\|\rho' - \tilde{\rho}\|_{\text{tr}} = \mathcal{O}\left(\zeta^{\frac{1}{4}}\right). \tag{25}$$

Let R be a diagonal matrix whose elements are $R_{ii} \in \left[-\frac{\sqrt{\zeta}}{n}, \frac{\sqrt{\zeta}}{n}\right]$ for $i \in [n]$, such that

$$W = \rho' + R,$$

and note that $R + \sqrt{\zeta} n^{-1} I \succeq 0$. Upon normalizing the trace, one can observe

$$\rho^* = \frac{1}{1+\sqrt{\zeta}}\left(\rho' + R + \sqrt{\zeta} n^{-1} I\right) \succeq 0,$$

with $\rho^*_{ii} = \frac{1}{n}n$ for all $i \in [n]$. Thus, $n\rho^*$ is a feasible solution to the SDO problem (3). Further, by a triangle inequality we have

$$\|\rho' - \rho^*\|_{\mathrm{tr}} = \frac{1}{1+\sqrt{\zeta}} \left\| \sqrt{\zeta}\rho' + R + \sqrt{\zeta}n^{-1}I \right\|_{\mathrm{tr}} = \mathcal{O}(\sqrt{\zeta}). \tag{26}$$

Combining equations (25) and (26) and noting $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$, applying another triangle inequality yields

$$\|\tilde{\rho} - \rho^*\|_{\mathrm{tr}} = \mathcal{O}\left(\zeta^{\frac{1}{4}}\right) = \mathcal{O}\left(\left[\left(\frac{\epsilon}{n\|C\|_F}\right)^4\right]^{\frac{1}{4}}\right) = \mathcal{O}\left(\frac{\epsilon}{n\|C\|_F}\right).$$

Then, the result follows from a matrix Hölder inequality:

$$|\mathrm{tr}\,(nC\rho) - \mathrm{tr}\,(nC\rho^*)| \leq n\|C\|\|\rho - \rho^*\|_{\mathrm{tr}} = \mathcal{O}\left(n\|C\|_F\zeta^{\frac{1}{4}}\right) = \mathcal{O}\left(n\|C\|_F\left[\frac{\epsilon}{n\|C\|_F}\right]\right) = \mathcal{O}\left(\epsilon\right).$$

□

5 Complexity

We now analyze the worst case overall running time of our Iterative Refinement Method given in Algorithm 2 in both the classical and quantum settings.

5.1 Classical running time

As we saw in Section 3, the complexity of using Algorithm 1 to solve the SDO problem (3) scales poorly in the inverse precision, with the classical algorithm exhibiting an $\mathcal{O}(\epsilon^{-12})$ dependence. In both the classical and quantum cases, our iterative refinement scheme reconciles the poor scaling in ϵ because it possesses the following two properties. First, we can obtain an arbitrarily precise solution to (5) in at most $\widetilde{\mathcal{O}}_{\frac{1}{\xi}}(1)$ iterations. Second, it suffices to treat ξ as fixed for the oracle calls that occur in each iteration, as the precision of the final solution is a byproduct of how we use these solution of the refining problems to produce a solution to (5).

The next result formalizes the above argument, and establishes the complexity of Algorithm 2 for the classical case.

Theorem 5. *Let $C \in \mathcal{S}^n$ with row sparsity s and $\epsilon \in (0,1)$. Then, fixing $\xi \in (0,1)$ with $0 < \epsilon \ll \xi < 1$, and setting $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$, a classical implementation of Algorithm 2 solves (3) up to additive error $\mathcal{O}(\epsilon)$ in time*

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\} \cdot \mathrm{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right).$$

The output of the algorithm is a classical description of a matrix $\tilde{\rho} \in \mathcal{S}^n_+$ that is a ζ-precise solution to (5). The entries of $\tilde{\rho}$ can be modified to construct a matrix ρ^* at trace distance $\mathcal{O}\left(\frac{\epsilon}{n\|C\|_F}\right)$ of $\tilde{\rho}$ in time $\mathcal{O}(n^2)$, such that $n\rho^*$ is a feasible point of the SDO problem (3).

Proof. Given that C is an s-sparse matrix, we can load C in $\mathcal{O}(ns)$ time, and from here we must compute $\|C\|_F$, which requires $\mathcal{O}(ns)$ arithmetic operations. In every iteration of Algorithm 2, we make a call to our subroutine in Algorithm 1, before updating the solution and preparing the next refining problem. Updating the solution involves matrix addition between two $n \times n$ matrices and requires $\mathcal{O}(n^2)$ arithmetic operations, whereas updating Q and ε for the next refining problem can be accomplished using $\mathcal{O}(n)$ arithmetic operations, as only the diagonal entries of Q need to be stored and maintained.

In view of Proposition 6, the dominant operation at each iteration is the use of Algorithm 1 to solve the SDO problem at hand. By Proposition 6, Algorithm 1 can be used to solve (14) to additive error ξ in time

$$\mathcal{T}_{HU}^{\text{classical}} = \mathcal{O}\left(\min\{n^2 s, n^\omega\} \log^2(n) \xi^{-3}\right).$$

If every call to Algorithm 1 is made using precision ξ^2, then by Corollary 4, Algorithm 2 converges in at most $\mathcal{O}\left(\log(\zeta^{-1})\right)$ iterations, and we can thus express the overall running time of Algorithm 2 as

$$\mathcal{O}\left(\left(\min\{n^2 s, n^\omega\} \log^2(n) \xi^{-6}\right) \log(\zeta^{-1})\right).$$

In the context of Algorithm 2, it suffices to carry out each of the calls to the SDO subroutine (calls to Algorithm 1) using fixed (i.e., constant) precision ξ to obtain a ζ-precise solution to (5). The above complexity thus reduces to

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\} \log^2(n) \log(\zeta^{-1})\right).$$

For our choice of $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$, one can observe

$$\mathcal{O}\left(\min\{n^2 s, n^\omega\} \log^2(n) \log(\zeta^{-1})\right) = \mathcal{O}\left(\min\{n^2 s, n^\omega\} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right).$$

Proposition 10 certifies that the above running time suffices to obtain a ρ from which we can construct ρ^* in time $\mathcal{O}(n^2)$, such that $n\rho^*$ is a feasible point of the SDO problem (3) satisfying

$$|\gamma_\zeta n\|C\|_F - \text{tr}\left(n\rho^* C\right)| = \mathcal{O}(\epsilon),$$

and the proof is complete. □

5.2 Quantum running time

Just as in the classical case, we show that a quantum implementation of Algorithm 2 mitigates the poor scaling in the running time with respect to the inverse precision.

Our quantum implementation of Algorithm 2 is provided in Algorithm 3. The relevant error parameters are the same as those appearing in Algorithm 2: *(i)* ξ, the fixed precision used to test closeness to the sets $\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}$ and $\mathcal{D}_{\eta\varepsilon}$ in every iteration, *(ii)* ζ, the precision to which the final solution solves (5), and *(iii)* ϵ, the additive error to which we seek to solve (3). In our initialization steps we set the values of Q, ε and η such that the first call to Algorithm 1 solves the original feasibility problem (10). We also create a vector $p = \mathbf{0}^{n\times 1}$ that will be used to maintain a classical description of the diagonal elements of our solution over the course of the algorithm.

At every iteration k, a call is made to Algorithm 1 with separation oracles $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$ to solve (14) using fixed precision ξ. If the oracles accept the candidate state, then Algorithm 1 returns a real-valued vector $y^{(k)} \in \mathbb{R}^2$ along with a diagonal matrix $D^{(k)}$ such that the Hamiltonian associated with the Gibbs state that solves the refining problem is

$$H^{(k)} = y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} D^{(k)},$$

with $\|y^{(k)}\|_1 \leq 4\log(n)\xi^{-2}$ and $\|D^{(k)}\| \leq 1$ for every $k \geq 0$. This allows us to efficiently describe the solution to each refining problem, and once the algorithm has terminated, it facilitates an efficient way to describe the final solution as well.[6] First, observe that the matrices $Q^{(k)}$ and $D^{(k)}$ can be completely described by their diagonal elements; letting $q^{(k)} \in \mathbb{R}^n$ and $d^{(k)} \in \mathbb{R}^n$ be the vectors that store the diagonal elements of $Q^{(k)}$ and $D^{(k)}$, respectively, we have

$$Q^{(k)} = (ee^\top - I) + \text{diag}\left(q^{(k)}\right),$$
$$D^{(k)} = \text{diag}\left(d^{(k)}\right).$$

[6]Requiring an explicit classical description of the solution would in fact lead to a worse running time overall when compared to the classical implementation we studied in Section 5.1.

Therefore, we store the solution to the refining problem at iteration k as the tuple

$$(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)}),$$

and the final solution to (5) is defined as

$$\tilde{\rho} = \sum_{k=0}^{K} \frac{1}{\eta^{(k)}(1+n\delta^{(k)})} \left[Q^{(k)} \circ \frac{\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)} + \delta^{(k)} I \right]. \tag{27}$$

We point out that this marks a key difference between the output of our algorithm and other quantum SDO solvers based on Gibbs sampling [10, 11, 12, 64, 65], which need only return a single state preparation pair. This however does not increase the cost of the method; the iteration bound in Corollary 4 ensures that there are only at most $\widetilde{\mathcal{O}}_{\frac{1}{\zeta}}(1)$ (i.e., a poly-logarithmic number) of these tuples to be stored over the course of the algorithm. Using the QRAM input model, one can use the stored tuples to construct a block-encoding of the final solution up to error θ using $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon},\frac{1}{\theta}}(\sqrt{n})$ queries to the QRAM and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}(n)$ classical operations. This construction, and the associated time complexity are analyzed later in Proposition 11. We further demonstrate that provided classical access to an s-sparse matrix $A \in \mathbb{R}^{n\times n}$ (with subnormalization factor 1) and access to QRAM, one can estimate $\operatorname{tr}(A\tilde{\rho})$ to additive error θ using $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{\sqrt{n}}{\theta}\right)$ queries to the QRAM and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}(ns)$ classical operations. If A has a subnormalization factor $\alpha_A > 1$, then θ must be scaled down by α_A, increasing the cost.

Additionally, we require Algorithm 1 to return the estimates $\tilde{p}^{(k)} \in \mathbb{R}^n$ (a classical estimate of the diagonal elements of the solution to the refining problem) and $\tilde{c}^{(k)} \in \mathbb{R}$ (a classical estimate of the objective value attained by the solution of the refining problem) that are used to test ξ-closeness for the accepted state. In this fashion, we can (classically) prepare the refining problem data for the next iteration without increasing the cost of the algorithm with respect to n; the objective value can be updated using $\mathcal{O}(1)$ arithmetic operations using $\tilde{c}^{(k)}$, while updating the residuals along the diagonal of ρ requires $\mathcal{O}(n)$ arithmetic operations provided classical access to $\tilde{p}^{(k)}$.

If the current solution is indistinguishable up to precision ζ from the maximally mixed state $n^{-1}I$, and provides an objective value of at least $\gamma - \zeta$, the algorithm terminates and reports the current solution. Otherwise, we construct the refining problem associated with our current solution and proceed to the next iteration.

The next result gives the overall running time required to solve (3) to additive error $\mathcal{O}(\epsilon)$ using the QRAM input model.

Theorem 6. *Let $C \in \mathcal{S}^n$, $\epsilon \in (0,1)$, and set $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$. Assume we have classical access to C. Then, in the QRAM input model, Algorithm 3 solves* (3) *up to additive error $\mathcal{O}(\epsilon)$ using*

$$\mathcal{O}\left(n^{1.5} \cdot \operatorname{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$$

accesses to the QRAM and $\mathcal{O}(ns)$ classical arithmetic operations.

The output of the algorithm is a collection of tuples $\{(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})\}_{k=0}^{K}$ such that

$$\tilde{\rho} = \sum_{k=0}^{K} \frac{1}{\eta^{(k)}\left(1+n\delta^{(k)}\right)} \left[Q^{(k)} \circ \widetilde{C} \frac{\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)} + \delta^{(k)} I \right] \succeq 0,$$

is a ζ-precise solution to (5)*. The entries of $\tilde{\rho}$ can be modified to construct a matrix ρ^* at trace distance $\mathcal{O}\left(\frac{\epsilon}{n\|C\|_F}\right)$ of $\tilde{\rho}$ in time $\mathcal{O}(n^2)$, such that $n\rho^*$ is a feasible point of the SDO problem* (3)*.*

Algorithm 3 Iterative Refinement for SDO Approximations of QUBOs using a quantum computer

Input: Error tolerances $\epsilon \in (0,1)$ and $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$, upper bound on objective value $\gamma \in [-1,1]$

Output: Tuples $\left\{\left(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)}\right)\right\}_{k=0}^{K}$ that define a ζ-precise solution $\tilde{\rho}$ to (5) using Equation (27)

Initialize: $\tilde{p}, \hat{p} \leftarrow \mathbf{0}^n$, $Q \leftarrow ee^\top$, $\varepsilon_i = \frac{1}{n}$ for $i \in [n]$, $\tilde{\gamma}, \hat{\gamma} \leftarrow 0$, $\eta^{(0)} \leftarrow 1$, $\delta^{(0)} \leftarrow 0$, $k \leftarrow 1$

$(y^{(0)}, D^{(0)}, \tilde{p}^{(0)}, \tilde{c}^{(0)}) \leftarrow$ solve (14) to precision $\frac{\xi^2}{16}$ using Algorithm 1 with oracles $O_{\mathcal{C}_{\eta(\gamma-\tilde{\gamma})}}$ and $O_{\mathcal{D}_{\eta\varepsilon}}$

$\tilde{\gamma}^{(0)} \leftarrow c^{(0)}$

$\varepsilon_i^{(0)} \leftarrow \tilde{p}_i^{(0)} - \frac{1}{n}$ for $i \in [n]$

$Q_{ii} \leftarrow \operatorname{sign}(-\varepsilon_i^{(0)})$ for $i \in [n]$

$\eta^{(1)} \leftarrow \frac{1}{\max\left\{\gamma - \tilde{\gamma}^{(0)}, \|\varepsilon^{(0)}\|_1\right\}}$

$\delta^{(1)} \leftarrow \frac{2}{n}\left(\|\varepsilon^{(0)}\|_1 + \frac{(\xi/4)^2}{\eta^{(1)}}\right)$

while $\max\{\gamma - \tilde{\gamma}, \|\varepsilon\|_1\} > \zeta$ **do**

1. Store refining problem data $\left(Q \circ \widetilde{C}, \eta^{(k)}\varepsilon^{(k-1)}, \eta^{(k)}\tilde{\gamma}^{(k-1)}\right)$
2. $(y^{(k)}, D^{(k)}, p^{(k)}, \tilde{c}^{(k)}) \leftarrow$ Solve (14) to precision $\frac{\xi^2}{16}$ using Algorithm 1
3. Update estimate of diagonal entries
$$\hat{p}_i^{(k)} \leftarrow \tilde{p}_i^{(k-1)} + \frac{Q_{ii}}{\eta^{(k)}} p_i^{(k)} \quad \text{for } i \in [n]$$
4. Apply spectrum shift to estimate of diagonal entries and update objective value
$$\tilde{p}_i^{(k)} \leftarrow \frac{1}{1+n\delta^{(k)}}\left(\hat{p}_i^{(k)} + \delta^{(k)}\right) \quad \text{for } i \in [n], \quad \tilde{\gamma}^{(k)} \leftarrow \tilde{\gamma}^{(k-1)} + \frac{1}{\eta^{(k)}}\left[\frac{1}{1+n\delta^{(k)}}\left(\tilde{c}^{(k)} + \delta^{(k)}\operatorname{tr}(\widetilde{C})\right)\right]$$
5. Store diagonal elements of Q and $D^{(k)}$ as the vectors $\left(q^{(k)}, d^{(k)}\right) \in \mathbb{R}^n \times \mathbb{R}^n$
6. Store description of solution to the refining problem $(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})$
7. Compute element-wise deviations from the maximally mixed state:
$$\varepsilon_i^{(k)} \leftarrow \tilde{p}_i^{(k)} - \frac{1}{n} \quad \text{for } i \in [n]$$
8. Classically update refining problem parameters:
$$Q_{ii} \leftarrow \operatorname{sign}\left(-\varepsilon_i^{(k)}\right) \text{ for } i \in [n], \quad \eta^{(k+1)} \leftarrow \frac{1}{\max\left\{\gamma - \tilde{\gamma}^{(k)}, \|\varepsilon^{(k)}\|_1\right\}}$$
9. Classically update spectrum shift parameter:
$$\delta^{(k+1)} \leftarrow \frac{2}{n}\left(\left\|\varepsilon^{(k)}\right\|_1 + \frac{(\xi/4)^2}{\eta^{(k+1)}}\right)$$
10. $k \leftarrow k+1$

end

Proof. Given that C is an s-sparse matrix, we can classically load C in $\mathcal{O}(ns)$ time. Similarly, for normalization purposes we classically compute $\|C\|_F$, which requires $\mathcal{O}(ns)$ arithmetic operations. In each iteration we use Algorithm 1 to solve (14), and use classical estimates of the diagonal elements of the refining solution, and a classical estimate of the objective value attained by the refining solution to update the solution and data for the refining problem we need to solve in the next iteration.

Let $\mathcal{T}_{HU}^{\text{quantum}}$ denote the cost of using Algorithm 1 as an approximate SDO subroutine at each iterate of Algorithm 3. By Proposition 8, Algorithm 1 solves (14) to additive error ξ using at most

$$\widetilde{\mathcal{O}}_{\frac{n}{\xi}}\left(n^{1.5}\xi^{-5}\right)$$

accesses to the QRAM. In the context of Algorithm 3, ξ is a fixed constant (and hence, so is ξ^2), so each ξ^2-precse oracle call to Algorithm 1 has an associated cost of

$$\mathcal{T}_{HU}^{\text{quantum}} = \widetilde{\mathcal{O}}_n\left(n^{1.5}\right)$$

accesses to the QRAM.

Classically updating the objective value requires $\mathcal{O}(1)$ arithmetic operations while updating the vector p which stores a classical description of the diagonal elements of our solution requires $\mathcal{O}(n)$ classical arithmetic operations. Similarly, accounting for the spectrum shift requires $\mathcal{O}(n)$ arithmetic operations; again this step is limited to operations on n-dimensional vectors and computing the trace of $\widetilde{C}$ (which can be stored once at the start of the algorithm). Likewise, ε and Q can each be updated using $\mathcal{O}(n)$ classical arithmetic operations, as we only need to store the diagonal elements of Q. This also implies that we can update $Q \circ \widetilde{C}$ using $\widetilde{\mathcal{O}}_n(n)$ operations, for only the diagonal elements need to be updated. When compared to loading and normalizing the coefficient matrix C, or our use of Algorithm 1 as a subroutine for solving (14), these intermediate computation steps are negligible and do not factor into the overall running time using $\mathcal{O}$ notation.

By Corollary 4, Algorithm 3 terminates in at most $\widetilde{\mathcal{O}}_{\frac{1}{\zeta}}(1)$ iterations. Therefore, the worst case complexity of Algorithm 3 can be bounded by

$$\mathcal{O}\left(n^{1.5} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$$

accesses to the QRAM, and $\mathcal{O}(ns)$ classical arithmetic operations. Just as in the proof of Theorem 5, applying Proposition 10 with our choice of $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$ implies that the above running time is sufficient to obtain a solution that can be used to solve (3) up to additive error $\mathcal{O}(\epsilon)$, and the proof is complete. □

We analyze the cost of Algorithm 3 without access to QRAM in Appendix A. Using the sparse-access input model, one can show that the resulting scheme exhibits an oracle complexity of

$$\mathcal{O}\left(n^{1.5}s^{0.5+o(1)} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right),$$

and requires $\mathcal{O}\left(n^{2.5}s^{0.5+o(1)} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$ additional gates. To summarize, in the absence of QRAM, the number of oracle accesses is a factor $\sqrt{s}$ larger due to the Hamiltonian simulation, and the gate complexity increases by a factor n due to the cost of constructing O_D without QRAM.

We conclude this section by establishing the costs of preparing a block-encoding of the final solution, and estimating trace inner products of the form $\text{tr}(A\tilde{\rho})$ for a given matrix A.

Proposition 11. *Suppose that Algorithm 3 is run with* $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$ *for some* $\epsilon \in (0,1)$*, and terminates after* K *iterations, classically outputting the tuples* $\left\{\left(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)}\right)\right\}_{k=0}^{K}$*. Then, letting*

$\widetilde{C} = C\|C\|_F^{-1}$ be stored in QRAM, and denoting the refining problem at iteration k by $\rho^{(k)}$, one can use $\{(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})\}_{k=0}^{K}$ to implement an $(n, \mathcal{O}(\log(n), \theta)$-block-encoding of

$$\tilde{\rho} = \sum_{k=0}^{K} \frac{1}{\eta^{(k)}\left(1+n\delta^{(k)}\right)} \left[Q^{(k)} \circ \frac{\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)} + \delta^{(k)} I \right],$$

with at most $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon},\frac{1}{\theta}}(\sqrt{n})$ queries to the QRAM and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon},\frac{1}{\theta}}(n)$ classical operations.

Proof. First, note that

$$\frac{\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)} = n^{-1} I$$

whenever $y = (0,0)^\top$. Thus, by choosing

$$\Delta := \sum_{k=1}^{K} \frac{\delta^{(k)}}{\eta^{(k)}\left(1+n\delta^{(k)}\right)},$$

and setting $y^{(K+1)} = (0,0)^\top$, $\delta^{(K+1)} = \frac{1}{n}$, $\eta^{(K+1)} = \frac{1}{n\Delta}$, and $Q^{(K+1)} = ee^\top$ we can simplify the expression of the final solution to

$$\tilde{\rho} = \sum_{k=0}^{K+1} \frac{1}{\eta^{(k)}\left(1+n\delta^{(k)}\right)} Q^{(k)} \circ \frac{\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)}.$$

To ensure that the stated complexity holds, for each $k \in [K+1]$, we block-encode

$$A^{(k)} = Q^{(k)} \circ \widetilde{C} + D^{(k)}.$$

First, note that with classical access to C and $q^{(k)}$, one can store $Q^{(k)} \circ \widetilde{C}$ in the QRAM by properly updating $\widetilde{C}$ in the QRAM. This step requires $\mathcal{O}(n)$ classical operations, as the only non-trivial computation that is performed is limited to the diagonal elements of the involved matrices. Then, with $Q^{(k)} \circ \widetilde{C}$ stored in QRAM, noting that $\left\|Q^{(k)} \circ \widetilde{C}\right\|_F \leq 1$ holds for every $k \in [K+1]$, we apply Lemma 3 to construct a $(1, \log(n)+2, \theta_1)$-block-encoding of $Q^{(k)} \circ \widetilde{C}$ in time $\mathcal{O}\left(\operatorname{polylog}\left(\frac{n}{\theta_1}\right)\right)$. Similarly, as we saw in the proof of Proposition 7, classical access to $d^{(k)}$ and access to QRAM implies one can implement a $(1, \log(n)+3, \theta_1)$-block-encoding of $D^{(k)}$ can be constructed in time $\widetilde{\mathcal{O}}_{\frac{n}{\theta_1}}(1)$.

Again following the proof of Proposition 7, applying Corollary 2 with $y^{(k)}$ satisfying $\|y^{(k)}\|_1 = \widetilde{\mathcal{O}}_n(\xi^{-1})$ implies that we can construct a unitary which prepares a copy of the Gibbs state $\rho^{(k)}$ encoding the solution to the refining problem at iteration k with at most

$$\widetilde{\mathcal{O}}_{\frac{n}{\xi}}\left(\sqrt{n}\alpha\xi^{-1}\right) = \widetilde{\mathcal{O}}_n\left(\sqrt{n}\right),$$

accesses to the QRAM, as $\alpha = 1$ and ξ is a fixed constant. Therefore, by Lemma 7, preparing a $(1, \log(n) + a, \theta_1)$ block-encoding of a purification of $\rho^{(k)}$ thus requires $\widetilde{\mathcal{O}}_{\frac{n}{\theta_1}}(\sqrt{n})$ queries to the QRAM.

Next, provided classical access to the vector $q^{(k)}$ that store the diagonal elements of $Q^{(k)}$, access to QRAM implies that we can efficiently implement an oracle $O_{Q^{(k)}}$ that returns the entries of $Q^{(k)}$ in a binary description:

$$O_{Q^{(k)}} : |i\rangle |j\rangle |0\rangle^{\otimes p} \mapsto |i\rangle |j\rangle \left|q_{ij}^{(k)}\right\rangle, \quad \forall i, j \in [2^{\log n}] - 1,$$

where $q_{ij}^{(k)}$ is a p-bit binary description of the ij-matrix element of $Q^{(k)}$ for $k = 0, \ldots, K+1$. By construction each matrix $Q^{(k)}$ may be fully dense, and hence an application of Lemma 4 with $s_r = s_c = n$ asserts that in the presence of QRAM, one can construct a $(n, \log(n) + 3, \theta_2)$-block-encoding of $Q^{(k)}$ in time $\widetilde{\mathcal{O}}_{\frac{n}{\theta_2}}(1)$.

From here, we can utilize Proposition 4 with $\theta_1 = \theta_2 = \frac{\tilde{\theta}}{10}$ to construct an $(n, a + 4\log(n^2) + 12, \tilde{\theta})$-block-encoding of $Q^{(k)} \circ \rho^{(k)}$ in time $\widetilde{\mathcal{O}}_{\frac{n}{\tilde{\theta}}}(1)$. Repeating the above steps for $k = 0, \ldots, K+1$, it follows that we can block-encode each of the terms $Q^{(k)} \circ \rho^{(k)}$ using at most

$$\widetilde{\mathcal{O}}_{n, \frac{1}{\tilde{\theta}}}\left(K\sqrt{n}\right) = \widetilde{\mathcal{O}}_{n, \|C\|_F, \frac{1}{\epsilon}, \frac{1}{\tilde{\theta}}}\left(\sqrt{n}\right)$$

queries to the QRAM and $\widetilde{\mathcal{O}}_{n, \|C\|_F, \frac{1}{\epsilon}, \frac{1}{\tilde{\theta}}}(n)$ classical operations, as $K = \mathcal{O}\left(\text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$ by Corollary 4.

Finally, what remains is to take the linear combination of these terms. To do so, we choose our weights to be $w_k = \frac{1}{2(1+n\delta^{(k)})\eta^{(k)}}$, which indeed satisfies $\|w\|_1 \leq 1$. Then, we can construct a $(K+2, \log(K+2), 0)$-state-preparation pair P_L, P_R for w, which can be constructed by taking a $\log(K+2)$-fold tensor product of the Hadamard gate, i.e.,

$$P_L = P_R = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes \log(K+2)}.$$

We are now in a position to apply Proposition 1, and choosing $\tilde{\theta} = \frac{\theta}{n}$, we can obtain W upon adding a control qubit to the circuits used to construct the block-encoding of each $Q^{(k)} \circ \rho^{(k)}$. As a result, we obtain an $(n, \mathcal{O}(\log(n), \theta)$-block-encoding of $\tilde{\rho}$ with a single use of W, P_R and $P_L^\dagger$. Summing the cost of each step in the construction we arrive at total cost of

$$\widetilde{\mathcal{O}}_{n, \|C\|_F, \frac{1}{\epsilon}, \frac{1}{\tilde{\theta}}}\left(\sqrt{n}\right)$$

queries to the QRAM and $\widetilde{\mathcal{O}}_{n, \|C\|_F, \frac{1}{\epsilon}, \frac{1}{\tilde{\theta}}}(n)$ classical operations, and proof is complete. □

Proposition 12. *Suppose that Algorithm 3 is run with* $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$ *for some* $\epsilon \in (0,1)$*, and terminates after* K *iterations, classically outputting the tuples* $\left\{\left(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)}\right)\right\}_{k=0}^{K}$*. Let* $A \in \mathbb{R}^{n \times n}$ *be a matrix with* $\|A\|_F \leq 1$ *and assume classical access to* A *and* $C/\|C\|_F$*. Then, with access to QRAM, one can compute a* θ*-precise estimate of* $\text{tr}(A\tilde{\rho})$ *using at most* $\widetilde{\mathcal{O}}_{n, \|C\|_F, \frac{1}{\epsilon}}\left(\frac{\sqrt{n}}{\theta}\right)$ *queries to the QRAM and* $\widetilde{\mathcal{O}}_{n, \|C\|_F, \frac{1}{\epsilon}}(n)$ *classical operations.*

Proof. See the proof of Theorem 8 in Appendix B. □

A QRAM-free version of Proposition 12 is also analyzed in Appendix B, and the cost is summarized in Corollary 5. Without access to QRAM, the cost increases with respect to n because computing the Hadamard product of block-encodings introduces n as a subnormalization factor. This is compounded in the running time, upon noting that we then have to scale down the error for the amplitude estimation steps by n, and constructing sparse-access oracles for the intermediate block-encodings of Q and D that arise in the trace estimation procedure requires $\widetilde{\mathcal{O}}_n(n)$ gates.

5.3 Comparison to existing SDO algorithms

Table 1 presents a comparison of the running time results for the algorithms we have proposed with the running times of the best performing methods from both the classical and quantum literature when applied to solving (3).

Note that when directly solving (3), $m = n$, and any feasible solution X to (3) satisfies $\text{tr}(X) = n$, implying $R = n$ for the algorithms based on the (Q)MMWU framework. We also point out that the running times in Table 1 take into account the role of sparsity in context of the algorithms, which is measured as the

References	Method	Runtime	Error Scaling
[35]	IPM	$\widetilde{\mathcal{O}}_{n,\frac{1}{\epsilon}}\left(n^{\omega+0.5}\right)$	ϵ
[7]	QIPM	$\widetilde{\mathcal{O}}_{n,\kappa,\frac{1}{\epsilon}}\left(\sqrt{n}(n^3\kappa\epsilon^{-1}+n^4)\right)$	ϵ
[42]	MMWU	$\widetilde{\mathcal{O}}_{n,\frac{1}{\epsilon}}\left(ns\epsilon^{-3.5}\right)$	$\|C\|_{\ell_1}\epsilon$
[64]	QMMWU	$\widetilde{\mathcal{O}}_{n,\frac{1}{\epsilon}}\left(n^{5.5}s\epsilon^{-4}\right)$	$n\|C\|\epsilon$
[11] (Classical)	HU	$\widetilde{\mathcal{O}}_{n,\|C\|}\left(\min\{n^2s,n^\omega\}\epsilon^{-12}\right)$	$n\|C\|\epsilon$
[11] (Quantum)	HU	$\widetilde{\mathcal{O}}_{n,\|C\|,\frac{1}{\epsilon}}\left(n^{2.5}s^{0.5+o(1)}\epsilon^{-28+o(1)}\exp\left(1.6\sqrt{\log(\epsilon^{-1})}\right)\right)$	$n\|C\|\epsilon$
[11] (Quantum)	HU-QRAM	$\widetilde{\mathcal{O}}_{n,\|C\|,\frac{1}{\epsilon}}\left(n^{1.5}s^{0.5+o(1)}\epsilon^{-28+o(1)}\exp\left(1.6\sqrt{\log(\epsilon^{-1})}\right)\right)$	$n\|C\|\epsilon$
This work (Classical)	IR-HU	$\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\min\{n^2s,n^\omega\}\right)$	ϵ
This work (Quantum)	IR-HU	$\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(n^{2.5}s^{0.5+o(1)}\right)$	ϵ
This work (Quantum)	IR-HU-QRAM	$\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}(n^{1.5})+ns$	ϵ

Table 1: Total running times for classical and quantum algorithms to solve (3).

maximum number of nonzero entries per row of the constraint matrices $A_1,\ldots,A_n$. When using either an IPM or CPM to solve (3), the n constraint matrices are $A_i = e_i e_i^\top$ (with row sparsity one) enforcing $X_{ii}=1$ for each diagonal element. On the other hand, algorithms based on the (Q)MMWU or HU frameworks solve (3) by reducing the problem to a feasibility problem; C enters into the resulting formulation as another constraint matrix, and as a result, the relevant sparsity parameter is the maximum number of non-zeroes per row of C, which we denote by s in Table 1.

There are additional considerations that need to be taken into account when making comparisons across methodologies listed in Table 1. Broadly speaking, both (Q)MMWUs and HU require normalizing the problem by an upper bound on the trace of a primal solution, and in the case of (3), we have the natural bound $\mathrm{tr}(X)=n$. Moreover, (Q)MMWUs and HU additionally normalize the cost matrix so that it exhibits unit norm with respect to some norm. While these modifications amount to scaling the optimal objective value of (3) by a fixed quantity, without employing any safeguards such as IR, these modifications impact the scaling of the error as reflected in the fourth column of Table 1. On the contrary, (Q)IPMs do not require the SDO problem to be normalized in any way. Finally there is a distinction with regard to output; (Q)IPMs explicitly report a classical description of the solution X, whereas only the classical HU algorithm of [11] and our own classical IR-HU method do so; the primal QMMWU of [64] reports a state-preparation pair y, and the MMWU algorithm found in [42] reports a "gradient" $G\in\mathcal{S}^n$ such that $X = W\exp(G)W$ for a diagonal matrix W. As we noted earlier, (Q)IPMs and (Q)MMWUs also utilize different definitions of optimality.

It can be easily seen that both the classical and quantum implementations of our proposed methodology outperform all existing algorithms that exhibit poly-logarithmic dependence on the precision ϵ. Our classical algorithm is only outperformed with respect to dimension by our own quantum algorithms, and the algorithm from [42], which has an exponentially worse dependence on the inverse prevision. Moreover, to achieve the same error scaling as our algorithms, the algorithm from [42] would require time $\widetilde{\mathcal{O}}_{n,\frac{1}{\epsilon}}\left(\|C\|_{\ell_1}^{3.5}ns\epsilon^{-3.5}\right)$. Up to poly-logarithmic factors, our quantum algorithms outperform each of the classical and quantum solvers in every parameter, suggesting the first evidence of quantum advantage for solving a special class of SDO problems. Moreover, our implementation with access to QRAM dominates all other algorithms. We therefore conclude that our proposed algorithms are respectively, the fastest both in the classical and quantum regimes.

6 Conclusion

In this work we devised an iterative refinement scheme for a particular class of semidefinite optimization problems. The key to our idea behind our speedup is to solve a sequence of related SDO problems in fixed

low precision, rather than solve one SDO problem using high accuracy requirements. Moreover, our solutions satisfy a far stronger approximation guarantee over previous quantum solution methodologies for this class of problem. We show that, provided access to QRAM, a quantum implementation of our algorithm can produce accurate solutions to SDO approximations of QUBO problems in time $\mathcal{O}\left(ns + n^{1.5} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$ in the worst case. In the absence of QRAM, one can bound the running time of the quantum algorithm using using the sparse-access input model, in which case the algorithm exhibits an oracle complexity of $\mathcal{O}\left(n^{2.5}s^{0.5+o(1)} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$. A classical implementation of the algorithm exhibits worst case running time of $\mathcal{O}\left(\min\{n^2 s, n^\omega\} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$, which is at least a $\sqrt{n}$ factor better than classical IPMs.

When compared to the best performing algorithms in the literature, our algorithms are the fastest in both the quantum and classical regimes, respectively. This work indicates that there could be a genuine quantum advantage (in the QRAM model) for this specific class of SDO problems; to establish such an advantage, one would have to show that no classical algorithm can beat the quantum running time. At the moment, we can only make the weaker claim that our quantum algorithm is faster than any currently known classical algorithm. We believe one can improve the theoretical performance of our classical algorithm by not explicitly computing the density operator in our subroutines. In particular, it may be possible to construct the separation oracles as we do in the quantum setting using techniques to classically estimate trace inner products of the form $\text{tr}(A\rho)$ (see, e.g., Appendix A in [65]), and applying ideas developed in [4, 42] to estimate the diagonal elements of matrix exponentials via randomized projection [37]. It remains an open question as to whether our techniques can be applied to general SDO problems using the matrix-multiplicative weights update framework as a subroutine.

Acknowledgements

The authors are grateful to David Gross and Richard Kueng, who pointed out an error in an earlier version of this paper. This project has been carried out thanks to funding by the Defense Advanced Research Projects Agency (DARPA), ONISQ grant W911NF2010022, titled The Quantum Computing Revolution and Optimization: Challenges and Opportunities.

A Running time of Algorithm 3 without QRAM

The following result from [11] gives the sample complexity of implementing the oracles in the sparse-access model.

Lemma 16 (see, proof of Lemma 3.3 in [11])**.** *We can implement the oracle $O_{\mathcal{C}_\gamma}$ on a quantum computer given access to $\mathcal{O}(\epsilon^{-2})$ copies of a state that is an $\frac{\epsilon}{8}$-approximation of the input state ρ in trace distance. The oracle $O_{\mathcal{D}_n}$ can be implemented using $\mathcal{O}(n\epsilon^{-2})$ $\frac{\epsilon}{8}$-approximate copies of the input, and the classical post-processing time needed to implement the oracle is $\mathcal{O}(n\epsilon^{-2})$.*

Next, we bound the overall complexity of Algorithm 1 without access to QRAM.

Proposition 13. *Suppose that $C \in \mathcal{S}^n$ has row sparsity s and $\xi \in (0, 1)$. Then, in the sparse-access input model, the complexity of solving* (5) *up to additive error ξ using Algorithm 1 on a quantum computer requires*

$$\widetilde{\mathcal{O}}_n\left(n^{1.5}\sqrt{s}^{1+o(1)}\xi^{-7+o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right)$$

queries to the input oracle O_C and $\widetilde{\mathcal{O}}_n\left(n^{2.5}\sqrt{s}^{1+o(1)}\xi^{-7+o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right)$ additional gates.

Proof. Our proof can be viewed as the QRAM-free analogue of the discussion found in [11, Section 3.4], and we repeat it here for completeness. In order to derive an appropriate bound on the per-iteration cost, we need to evaluate the cost of constructing our separation oracles. By Lemma 16, we can conclude that the

time to construct the oracle $O_{\mathcal{D}_n}$ for the diagonal elements dominates that of constructing the oracle $O_{\mathcal{C}_\gamma}$ to test the objective value.

We now turn our attention to the cost of simulating our Hamiltonian H. From the results in [57, Appendix] it follows that we can produce a state that is $\frac{\xi}{8}$ close to ρ using $\widetilde{\mathcal{O}}(\sqrt{n}\xi^{-3})$ invocations of a controlled U which satisfies

$$\left\|U - e^{it_0 H}\right\| \leq \mathcal{O}\left(\xi^3\right),$$

with $t_0 = \frac{\pi}{4\|H\|}$. Further, the authors in [11] note that each of the Hamiltonians we seek to simulate are of the form $H = y_1 C\|C\|_F^{-1} + y_2 D$ where $y_1, y_2 = \mathcal{O}(\log(n)\xi^{-1})$ and D is a diagonal matrix which satisfies $\|D\| \leq 1$. Invoking [16, Theorem 1], we can simulate H for time t up to error ξ^3 using

$$\widetilde{\mathcal{O}}\left(t(a+b)\exp\left(1.6\sqrt{\log\left(\log(n)t\xi^{-3}\right)}\right)\right)$$

separate simulations of $y_1 C\|C\|_F$ and $y_2 D$.

As noted in [11], access to the oracles O_{sparse} and O_C we described in Section 2.1.1 allows us to simulate $\exp(it\widetilde{C})$ in time $\mathcal{O}\left((t\sqrt{s})^{1+o(1)}\xi^{o(1)}\right)$ if we utilize the algorithm in [45]. Similarly, we follow [11] in constructing an oracle O_D acting on $\mathbb{C} \otimes (\mathbb{C}^2)^{\otimes a}$, where a is a sufficiently large constant such that we can represent the diagonal elements of D as

$$O_D\left|i, z\right\rangle \mapsto \left|i, z \oplus D_{ii}\right\rangle$$

to the desired level of precision in binary. Accordingly, we can simulate e^{iDt} for $t = \widetilde{\mathcal{O}}(\xi^{-1})$ using $\widetilde{\mathcal{O}}_n(1)$ queries to O_D and $\widetilde{\mathcal{O}}_n(1)$ elementary operations [8], and we can implement O_D using $\widetilde{\mathcal{O}}_n(n)$ gates.

To summarize, the Gibbs sampler from [57] requires $\widetilde{\mathcal{O}}(\sqrt{n}\xi^{-3})$ Hamiltonian simulation steps, each of which requires time

$$\widetilde{\mathcal{O}}\left(\sqrt{s}^{1+o(1)}\xi^{o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right).$$

Hence, each iteration of Algorithm 1 requires a total of

$$\widetilde{\mathcal{O}}_n\left(n^{1.5}\sqrt{s}^{1+o(1)}\xi^{-5+o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right)$$

sparse-access oracle queries. Combining the above per-iteration cost with the iteration bound $\mathcal{O}(\log(n)\xi^{-2})$ provided in Theorem 3, it follows that Algorithm 1 solves (5) up to additive error ξ with at most

$$\widetilde{\mathcal{O}}_n\left(n^{1.5}\sqrt{s}^{1+o(1)}\xi^{-7+o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right)$$

queries to the input oracle O_C and $\widetilde{\mathcal{O}}_n\left(n^{2.5}\sqrt{s}^{1+o(1)}\xi^{-7+o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right)$ additional gates. □

Theorem 7 formalizes the complexity of of Algorithm 3 in the quantum setting without access to QRAM. In our analysis, we employ the same Hamiltonian simulation subroutines and Gibbs sampler used in [11] to construct our separation oracles.

Theorem 7. *Let $C \in \mathcal{S}^n$ with row sparsity s and $\epsilon \in (0,1)$. Then, setting $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$ and fixing $\xi = 10^{-2}$, a quantum implementation of Algorithm 3 using the sparse-access input model solves* (3) *up to additive error $\mathcal{O}(\epsilon)$ using*

$$\mathcal{O}\left(n^{1.5}s^{0.5+o(1)} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$$

queries to the input oracle O_C and $\mathcal{O}\left(n^{2.5}s^{0.5+o(1)} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$ additional gates.

The output of the algorithm is a collection of tuples $\{(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})\}_{k=0}^K$ such that

$$\tilde{\rho} = \sum_{k=0}^{K} \frac{1}{\eta^{(k)}(1+n\delta^{(k)})}\left[Q^{(k)} \circ \frac{\exp\left(-\left[y_1^{(k)}Q^{(k)} \circ \widetilde{C} + y_2^{(k)}\,\text{diag}\left(d^{(k)}\right)\right]\right)}{\text{tr}\left(\exp\left(-\left[y_1^{(k)}Q^{(k)} \circ \widetilde{C} + y_2^{(k)}\,\text{diag}\left(d^{(k)}\right)\right]\right)\right)} + \delta^{(k)}I\right] \succeq 0,$$

is a ζ-precise solution to (5)*. The entries of $\tilde{\rho}$ can be modified to construct a matrix ρ^* at trace distance $\mathcal{O}\left(\frac{\epsilon}{n\|C\|_F}\right)$ of $\tilde{\rho}$ in time $\mathcal{O}(n^2)$, such that $n\rho^*$ is a feasible point of the SDO problem* (3)*.*

Proof. Given that C is an s-sparse matrix, we can load C in $\mathcal{O}(ns)$ time. Similarly, for normalization purposes we classically compute $\|C\|_F$, which requires $\mathcal{O}(ns)$ arithmetic operations. In each iteration we use Algorithm 1 to solve (14), and use classical estimates of the diagonal elements of the refining solution, and a classical estimate of the objective value attained by the refining solution to update the solution and data for the refining problem we need to solve in the next iteration.

Letting $\mathcal{T}_{HU}^{\text{sparse}}$ be the cost of using Algorithm 1 as an approximate SDO subroutine, we saw in Proposition 13, Algorithm 1 solves (14) to additive error ξ using

$$\mathcal{T}_{HU}^{\text{sparse}} = \widetilde{\mathcal{O}}_n\left(n^{1.5}\sqrt{s}^{1+o(1)}\xi^{-7+o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right)$$

queries to the oracle describing the problem data and $\widetilde{\mathcal{O}}_n\left(n^{2.5}\sqrt{s}^{1+o(1)}\xi^{-7+o(1)}\exp\left(1.6\sqrt{\log(\xi^{-1})}\right)\right)$ additional gates. In the context of Algorithm 3, ξ is a fixed constant, so the cost of our oracle call to Algorithm 1 simplifies to

$$\mathcal{T}_{HU}^{\text{sparse}} = \widetilde{\mathcal{O}}_n\left(n^{1.5}\sqrt{s}^{1+o(1)}\right)$$

queries to the oracle describing the problem data and $\widetilde{\mathcal{O}}_n\left(n^{2.5}\sqrt{s}^{1+o(1)}\right)$ additional gates.

Classically updating the objective value requires $\mathcal{O}(1)$ arithmetic operations while updating the vector p which stores a classical description of the diagonal elements of our solution as

$$p_i \leftarrow p_i + \frac{Q_{ii}}{\eta^{(k)}}\tilde{p}_i^{(k)}$$

requires $\mathcal{O}(n)$ arithmetic operations. Again, ε and Q can each be updated using $\mathcal{O}(n)$ arithmetic operations, as we only need to store the diagonal elements of Q. This also implies that we can also calculate $Q \circ \widetilde{C}$ in time $\mathcal{O}(n)$, for only the element-wise products along the diagonal are non-trivial. When compared to loading and normalizing the data or our use of Algorithm 1 as a subroutine for solving (14), these intermediate computation steps are negligible and do not factor into the overall running time using $\mathcal{O}$ notation.

Factoring in the $\mathcal{O}\left(\text{polylog}\left(\frac{1}{\zeta}\right)\right) = \mathcal{O}\left(\text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$ from Corollary 4, it follows that a quantum implementation of Algorithm 3 requires at most

$$\mathcal{O}\left(n^{1.5}s^{0.5+o(1)} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$$

queries to the input oracle O_C and $\mathcal{O}\left(n^{2.5}s^{0.5+o(1)} \cdot \text{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$ additional gates. Just as in the proof of Theorem 5, applying Proposition 10 with our choice of $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$ implies that the above running time is sufficient to obtain a solution that can be used to solve (3) up to additive error $\mathcal{O}(\epsilon)$, and the proof is complete. □

B Estimating trace inner products with the final solution

Given that we do not explicitly report a classical description of the final solution $\tilde{\rho}$ defined in equation (27), it may be of interest to understand how, for a user specified matrix A, one can compute the trace inner product $\text{tr}(A\tilde{\rho})$. We outline a procedure for doing so using the state preparation pair description of solution $\{(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})\}_{k=0}^{K}$ in Algorithm 4, and subsequently analyze the complexity of doing so.

Theorem 8. *Let $A \in \mathbb{R}^{n\times n}$, and $\widetilde{C} \in \mathcal{S}^n$ be stored in QRAM, $\theta \in (0,1)$, and $\{(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})\}_{k=0}^{K}$ be a state preparation pair description of the solution obtained from running Algorithm 3 to final precision*

Algorithm 4 Trace estimation procedure for the final solution

Input: Access to an s-sparse matrix $A \in \mathbb{R}^{n\times n}$ with $\|A\|_F \leq 1$, state preparation pair description of solution $\{(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})\}_{k=0}^{K}$, precision $\theta \in (0,1)$, $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$

Output: A θ-precise classical estimate of $\mathrm{tr}(A\tilde{\rho})$

Initialize: $a \leftarrow 0$, $k \leftarrow 0$, $y^{(K+1)} \leftarrow (0,0)^\top$, $\delta^{(K+1)} = 0$, $\eta^{(K+1)} = \frac{\sum_{k\in[K]} \eta^{(k)}(1+n\delta^{(k)})}{n\sum_{k\in[K]}\delta^{(k)}}$, $Q^{(K+1)} \leftarrow ee^\top$

for $k = 0, \ldots, K+1$ **do**

1. Implement an $(\alpha, a, \zeta/2(K+2))$-block-encoding of $Q^{(k)} \circ A$
2. Use block-encoding of $Q^{(k)} \circ A$ to implement a trace estimator for
$$a^{(k)} = \mathrm{tr}\left[\left(Q^{(k)} \circ A\right)\left(\frac{\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \mathrm{diag}\left(d^{(k)}\right)\right]\right)}{\mathrm{tr}\left(\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \mathrm{diag}\left(d^{(k)}\right)\right]\right)\right)}\right)\right]$$
3. Use $\mathcal{O}\left(\frac{K}{\theta}\right)$ samples from the trace estimator to produce $\frac{\theta}{K+2}$-precise estimate $\tilde{a}^{(k)}$ of $a^{(k)}$
4. Update solution:
$$a \leftarrow a + \frac{1}{\eta^{(k)}(1+n\delta^{(k)})}\tilde{a}^{(k)}$$
5. $k \leftarrow k+1$

end

$\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$. *Suppose A is an s-sparse matrix with $\|A\|_F \leq 1$, and assume classical access to A and $\widetilde{C} \in \mathcal{S}^n$. Then, Algorithm 4 outputs a θ-precise estimate of* $\mathrm{tr}(A\tilde{\rho})$ *using at most*

$$\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{\sqrt{n}}{\theta}\right)$$

queries to the QRAM and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}(ns)$ classical operations.

Proof. We begin by establishing the correctness of Algorithm 4. First, note that following the proof of Proposition 11, we can simplify the expression of the final solution to

$$\tilde{\rho} = \left[\sum_{k=0}^{K+1} \frac{1}{\eta^{(k)}(1+n\delta^{(k)})} Q^{(k)} \circ \frac{\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \mathrm{diag}\left(d^{(k)}\right)\right]\right)}{\mathrm{tr}\left(\exp\left(-\left[y_1^{(k)} Q^{(k)} \circ \widetilde{C} + y_2^{(k)} \mathrm{diag}\left(d^{(k)}\right)\right]\right)\right)}\right].$$

by setting $y^{(K+1)} = (0,0)^\top$, $\delta^{(K+1)} = 0$, $\eta^{(K+1)} = \frac{1}{n}\sum_{k\in[K]} \frac{\eta^{(k)}(1+n\delta^{(k)})}{\delta^{(k)}}$, and $Q^{(K+1)} = ee^\top$. Then, by

linearity of the trace and Lemma 1, one has:

$$\begin{aligned}
\operatorname{tr}(A\tilde{\rho}) &= \operatorname{tr}\left(A\left[\sum_{k=0}^{K+1}\frac{1}{\eta^{(k)}(1+n\delta^{(k)})}Q^{(k)}\circ\frac{\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)}\right]\right)\\
&= \sum_{k=0}^{K+1}\frac{1}{\eta^{(k)}(1+n\delta^{(k)})}\operatorname{tr}\left(A\left[Q^{(k)}\circ\frac{\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)}\right]\right)\\
&= \sum_{k=0}^{K+1}\frac{1}{\eta^{(k)}(1+n\delta^{(k)})}\operatorname{tr}\left(\left(Q^{(k)}\circ A\right)\frac{\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)}\right).
\end{aligned}$$

In other words, the output of Algorithm 4 is indeed an estimate of $\operatorname{tr}(A\tilde{\rho})$.

Next, we analyze the complexity of the procedure. If A is classically known, one can store $Q^{(k)}\circ A$ in the QRAM using $\mathcal{O}(ns)$ classical operations, as A is s-sparse. With $Q\circ A$ stored in a QRAM data structure, one can apply Lemma 3 to implement an $(1,\log(n)+2,\zeta/2(K+2))$-block-encoding of $Q\circ A$ in time $\widetilde{\mathcal{O}}_{\frac{nK}{\zeta}}(1)$ (as $\|Q\circ A\|_F\leq\|A\|_F\leq 1$ for any Q defined according to (13)). As we saw in the proof of Proposition 11, with $\widetilde{C}$ stored in QRAM, one can implement the state

$$\rho^{(k)}=\frac{\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)}{\operatorname{tr}\left(\exp\left(-\left[y_1^{(k)}Q^{(k)}\circ\widetilde{C}+y_2^{(k)}\operatorname{diag}\left(d^{(k)}\right)\right]\right)\right)}$$

using at most

$$\widetilde{\mathcal{O}}_n\left(\sqrt{n}\right),$$

accesses to the QRAM and $\mathcal{O}(n)$ classical operations.

Having prepared the state $\rho^{(k)}$ and a $(1,\log(n)+2,\zeta/2(K+2))$-block-encoding U_k of $Q^{(k)}\circ A$, Lemma 8 asserts that one can implement a trace estimator for

$$\operatorname{tr}\left[\left(Q^{(k)}\circ A\right)\rho^{(k)}\right]$$

with bias at most $\frac{\zeta}{K+2}$ using $\widetilde{\mathcal{O}}(1)$ applications of U_k and $U_k^\dagger$. Applying amplitude estimation using $\mathcal{O}\left(\frac{K}{\theta}\right)=\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{1}{\theta}\right)$ samples from the estimator, we obtain a $\frac{\theta}{K+2}$-precise classical estimate $\tilde{a}^{(k)}$ of $a^{(k)}$, as $K=\mathcal{O}\left(\operatorname{polylog}\left(n,\|C\|_F,\frac{1}{\epsilon}\right)\right)$.

From here, we classically update a using $\mathcal{O}(1)$ arithmetic operations. Therefore, each iteration of Algorithm 4 requires at most

$$\widetilde{\mathcal{O}}_{n,\frac{K}{\zeta}}\left(\frac{\sqrt{n}}{\theta}\right)$$

accesses to the QRAM and $\mathcal{O}(ns)$ classical operations. Summing over $K+2$ iterations implies a total of

$$\widetilde{\mathcal{O}}_{n,\frac{K}{\zeta}}\left(K\left(\frac{\sqrt{n}}{\theta}\right)\right)=\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{\sqrt{n}}{\theta}\right)$$

accesses to the QRAM and

$$\mathcal{O}\left(Kns\right)=\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(ns\right)$$

classical operations. The proof is complete. □

Note that if $\|A\|_F>1$, because of the subnormalization to block-encode A we need to increase precision of the estimation procedure: the cost increases by a factor proportional to $\|A\|_F$.

Corollary 5. *Let $A \in \mathbb{R}^{n\times n}$, $\theta \in (0,1)$, and $\{(\eta^{(k)}, y^{(k)}, q^{(k)}, d^{(k)}, \delta^{(k)})\}_{k=0}^{K}$ be a state preparation pair description of the solution obtained from running Algorithm 3 to final precision $\zeta = \left(\frac{\epsilon}{n\|C\|_F}\right)^4$. Suppose A is an s-sparse matrix with $\|A\|_F \leq 1$, and assume sparse oracle access to A and $\widetilde{C} \in \mathcal{S}^n$. Then, Algorithm 4 outputs a θ-precise estimate of $\mathrm{tr}(A\tilde{\rho})$ using at most*

$$\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{n^{2.5}s^2}{\theta}\right)$$

queries to O_A, O_C, and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{n^{3.5}s^2}{\theta}\right)$ additional gates.

Proof. Provided classical access to A, we use Lemma 4 with $s_r = s_c$ to construct an $(s, \log(n)+3, \theta/n)$-block-encoding of A with two uses of O_A (an oracle describing the elements of A in binary), and additionally using $\widetilde{\mathcal{O}}_n(1)$ one and two qubit gates.

Likewise, with access to the oracle O_C describing the elements of $\widetilde{C}$, one can construct an $(s, \log(n)+3, \theta/n)$-block-encoding of $\widetilde{C}$ with two uses of O_C, and additionally using $\widetilde{\mathcal{O}}_n(1)$ one and two qubit gates. Note that without access to QRAM, we must compute the Hadamard products by taking the Hadamard products of block-encodings, which causes the subnormalization factor for the Hadamard product $Q^{(k)} \circ \widetilde{C}$ to be ns, as $Q^{(k)}$ may be fully dense and C is s-sparse. It follows that preparing one copy of each Gibbs state requires

$$\widetilde{\mathcal{O}}_n\left(\sqrt{n}(ns)\right) = \widetilde{\mathcal{O}}_n\left(n^{1.5}s\right)$$

accesses to block-encodings of $Q^{(k)} \circ \widetilde{C}$ and D, which each require an additional $\widetilde{\mathcal{O}}_n(n)$ gates (to construct sparse-access oracles for $Q^{(k)}$ and D).

Similarly, the subnormalization factor for a block-encoding U_k of $Q^{(k)} \circ A$ will be ns. Having prepared the state $\rho^{(k)}$ and a block-encoding $Q^{(k)} \circ A$, Lemma 8 asserts that one can implement a trace estimator for

$$\mathrm{tr}\left[\left(Q^{(k)} \circ A\right)\rho^{(k)}\right]$$

with bias at most $\frac{\zeta}{K+2}$ using $\widetilde{\mathcal{O}}(ns)$ applications of U_k and $U_k^\dagger$. Applying amplitude estimation using $\mathcal{O}\left(\frac{K}{\theta}\right) = \widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{1}{\theta}\right)$ samples from the estimator to obtain a $\frac{\theta}{K+2}$-precise classical estimate $\tilde{a}^{(k)}$ of $a^{(k)}$, as $K = \mathcal{O}\left(\mathrm{polylog}\left(n, \|C\|_F, \frac{1}{\epsilon}\right)\right)$.

Just as in the QRAM setting, classically updating a requires $\mathcal{O}(1)$ arithmetic operations. Therefore, without access to QRAM, each iteration of Algorithm 4 requires at most

$$\widetilde{\mathcal{O}}_{n,\frac{K}{\zeta}}\left(\frac{n^{2.5}s^2}{\theta}\right) = \widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{n^{2.5}s^2}{\theta}\right)$$

applications of block-encodings for $Q^{(k)} \circ \widetilde{C}$, $D^{(k)}$ and $Q^{(k)} \circ A$ and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{n^{3.5}s^2}{\theta}\right)$ additional gates. This corresponds to $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{n^{2.5}s^2}{\theta}\right)$ queries to O_A and O_C in each iteration, and $\widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}\left(\frac{n^{3.5}s^2}{\theta}\right)$ additional gates. Summing over the $K+2 = \widetilde{\mathcal{O}}_{n,\|C\|_F,\frac{1}{\epsilon}}(1)$ iterations yields the stated complexity. □

References

[1] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[2] Noga Alon, W. Fernandez De La Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of MAX-CSP Problems. *Journal of Computer and System Sciences*, 67(2):212–243, 2003.

[3] David L. Applegate, William Cook, Sanjeeb Dash, and Daniel G. Espinoza. Exact solutions to linear programming problems. *Operations Research Letters*, 35(6):693–699, 2007.

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

[5] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. *Journal of the ACM (JACM)*, 63(2):1–35, 2016.

[6] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010, 2015.

[7] Brandon Augustino, Giacomo Nannicini, Tamás Terlaky, and Luis F. Zuluaga. Quantum interior point methods for semidefinite optimization. *arXiv preprint arXiv:2112.06025*, 2021.

[8] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse Hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[9] Rajendra Bhatia. *Matrix Analysis*, volume 169. Springer Science & Business Media, 2013.

[10] 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 Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 27:1–27:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[11] Fernando G.S.L. Brandão, Richard Kueng, and Daniel Stilck França. Faster quantum and classical SDP approximations for quadratic binary optimization. *Quantum*, 6:625, 2022.

[12] Fernando G.S.L. Brandão and Krysta M. Svore. Quantum speed-ups for solving semidefinite programs. In Rafail Ostrovsky and Chris Umans, editors, *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–426. IEEE, 2017.

[13] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: improved regression techniques via faster Hamiltonian simulation. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132, pages 33:1–33:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[14] Chen-Fu Chiang, Anirban Chowdhury, and Pawel Wocjan. Space-efficient quantization method for reversible markov chains. *arXiv preprint arXiv:2206.06886*, 2022.

[15] 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.

[16] Andrew M. Childs and Nathan Wiebe. Hamiltonian Simulation using linear combinations of unitary operations. *Quantum Information and Computation*, 12(11–12):901–924, Nov 2012.

[17] Anirban Narayan Chowdhury and Rolando D. Somma. Quantum algorithms for Gibbs sampling and Hitting-Time estimation. *Quantum Information & Computing*, 17(1–2):41–64, Feb 2017.

[18] William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter. An exact rational mixed-integer programming solver. In Oktay Günlük and Gerhard J. Woeginger, editors, *International Conference on Integer Programming and Combinatorial Optimization*, pages 104–116. Springer, 2011.

[19] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[20] Aleta Berk Finnila, Maria A. Gomez, C. Sebenik, Catherine Stenson, and Jimmie D. Doll. Quantum annealing: A new method for minimizing multidimensional functions. *Chemical Physics Letters*, 219(5-6):343–348, 1994.

[21] Daniel Stilck França. Perfect sampling for quantum Gibbs states. *Quantum Information and Computation*, 18:361–388, 2018.

[22] Alan Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[23] András Gilyén. *Quantum singular value transformation & its algorithmic applications*. PhD thesis, University of Amsterdam, 2019.

[24] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 193–204, 2019.

[25] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, 2008.

[26] Ambros M. Gleixner and Daniel E. Steffy. Linear programming using limited-precision oracles. *Mathematical Programming*, 183(1):525–554, 2020.

[27] Ambros M. Gleixner, Daniel E. Steffy, and Kati Wolter. Improving the accuracy of linear programming solvers with iterative refinement. In Joris van der Hoeven and Mark van Hoeij, editors, *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, pages 187–194, 2012.

[28] Ambros M. Gleixner, Daniel E. Steffy, and Kati Wolter. Iterative refinement for linear programming. *INFORMS Journal on Computing*, 28(3):449–464, 2016.

[29] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[30] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 2013.

[31] Sander Gribling. *Applications of optimization to factorization ranks and quantum information theory*. PhD thesis, Tilburg University, 2019.

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

[33] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[34] Roger Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press Cambridge, UK, 1994.

[35] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In Sandy Irani, Lisa O'Conner, and Patrick Kellenberger, editors, *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 910–918. IEEE, 2020.

[36] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 944–953, 2020.

[37] William B. Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[38] Michael J. Kastoryano and Fernando G.S.L. Brandao. Quantum Gibbs samplers: The commuting case. *Communications in Mathematical Physics*, 344(3):915–957, 2016.

[39] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Physical Review A*, 101(2):022316, 2020.

[40] Iordanis Kerenidis and Anupam Prakash. A quantum interior point method for LPs and SDPs. *ACM Transactions on Quantum Computing*, 1(1):1–32, 2020.

[41] Christopher King. Inequalities for trace norms of 2×2 block matrices. *Communications in Mathematical Physics*, 242(3):531–545, 2003.

[42] Yin Tat Lee and Swati Padmanabhan. An $\widetilde{\mathcal{O}}(m/\varepsilon^{3.5})$-cost algorithm for semidefinite programs with diagonal constraints. In Jacob Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory*, pages 3069–3119. PMLR, 2020.

[43] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In Rafail Ostrovsky and Venkatesan Guruswami, editors, *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1049–1065. IEEE, 2015.

[44] László Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25(1):1–7, 1979.

[45] Guang Hao Low. Hamiltonian simulation with nearly optimal dependence on spectral norm. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 491–502, 2019.

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

[47] Renato D.C. Monteiro. Polynomial convergence of primal-dual algorithms for semidefinite programming based on the Monteiro and Zhang family of directions. *SIAM Journal on Optimization*, 8(3):797–812, 1998.

[48] Arkadi Nemirovskii. Efficient methods for large-scale convex optimization problems. *Ekonomika i Matematicheskie Metody*, 15(1), 1979.

[49] Arkadi Nemirovskii and David B. Yudin. Problem complexity and method efficiency in optimization. 1983.

[50] Yurii E. Nesterov and Arkadi Nemirovskii. A general approach to polynomial-time algorithms design for convex programming. *Report, Central Economical and Mathematical Institute, USSR Academy of Sciences, Moscow*, 1988.

[51] Yurii E. Nesterov and Arkadi Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13. SIAM, 1995.

[52] Yurii E. Nesterov and Michael J. Todd. Self-scaled barriers and interior-point methods for convex programming. *Mathematics of Operations Research*, 22(1):1–42, 1997.

[53] Yurii E. Nesterov and Michael J. Todd. Primal-dual interior-point methods for self-scaled cones. *SIAM Journal on Optimization*, 8(2):324–364, 1998.

[54] Michael A. Nielsen and Isaac Chuang. *Quantum Computation and Quantum Information*. American Association of Physics Teachers, 2002.

[55] Foad Mahdavi Pajouh, Balabhaskar Balasundaram, and Oleg A. Prokopyev. On characterization of maximal independent sets via quadratic optimization. *Journal of Heuristics*, 19(4):629–644, 2013.

[56] Gábor Pataki and Aleksandr Touzov. How do exponential size solutions arise in semidefinite programming? *arXiv preprint arXiv:2103.00041*, 2021.

[57] 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.

[58] Maurice Sion. On general minimax theorems. *Pacific Journal of Mathematics*, 8(1):171–176, 1958.

[59] Jos F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.

[60] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. SDPT3—a MATLAB software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

[61] 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(Jun):995–1018, 2005.

[62] Joran van Apeldoorn. *A quantum view on convex optimization*. PhD thesis, University of Amsterdam, February 2020.

[63] Joran van Apeldoorn. Quantum probability oracles & multidimensional amplitude estimation. In Min-Hsiu Hsieh, editor, *16th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[64] Joran van Apeldoorn and András Gilyén. Improvements in Quantum SDP-Solving with Applications. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[65] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum SDP-solvers: Better upper and lower bounds. *Quantum*, 4:230, 2020.

[66] James Hardy Wilkinson. *Rounding Errors in Algebraic Processes*. Courier Corporation, 1994.

[67] Man-Hong Yung and Alán Aspuru-Guzik. A quantum–quantum Metropolis algorithm. *Proceedings of the National Academy of Sciences*, 109(3):754–759, 2012.