Quantum Algorithms for Semidefinite Programming and its Applications

Asish Kumar Mandoi* Arya Bhatta† Jignesh Mohanty‡
Indian Institute of Technology Kanpur

June 20, 2022

Abstract

In this report, we take a detailed look at Arora and Kale's algorithm which uses a matrix version of the "multiplicative weights update" method for solving semidefinite programs (SDP). We also see how it is different from other classical methods. Brandão and Svore were the first to modify the structure of the Arora-Kale framework and propose a quantum algorithm with quadratic speed-ups in terms of the number of dimensions and constraints of the SDP. We identify the specific part of the Arora-Kale algorithm which Brandão and Svore *quantized* achieving a quantum speed-up. Brandão and Svore's work has inspired further improvements by Apeldoorn, Gilyèn, et al. and even resulted in an optimal dependence on several parameters of an SDP. We analyze and compare the running time complexities of these algorithms. Finally, we also look at how these algorithms can be applied in specific problems that can be formulated as an SDP like Shadow Tomography and Quantum Error Recovery.

1. Introduction

1.1. Semidefinite Programs

Semidefinite programs (SDPs) are a fundamental class of convex optimization problems that have become an important tool for designing efficient optimization and approximation algorithms. Today, they have applications in a wide range of disciplines of engineering, combinatorial optimization problems, control theory, information theory among others. The primary reason for their wide applicability is because they encompass non-linear, but convex, constraints forming a generalized version of standard problems like Linear and Quadratic Programs, and at the same time require methods of polynomial time complexity, similar to that of Linear Programs, to be solved.

We consider the following primal form of an SDP:

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

and the following dual form:

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

Several methods have been used for solving SDPs out of which algorithms based the interior point methods are regarded as the state-of-the-art high accuracy algorithms. However, if we can tolerate large enough error, faster algorithms based on the Multiplicative Weights Update Method can be obtained. In this report, we shall focus on this method developed by Arora & Kale [3]. Their algorithm has been modified by Brandão & Svore to achieve polynomial speed-up compared to existing state-of-the-art classical SDP solvers using quantum algorithms, and further refined by Apeldoorn, Gilyén, et al.

1.2. Quantum Algorithms

Quantum Computing is an emerging field that leverages Quantum Mechanical properties such as superposition and entanglement to perform calculations that otherwise take exponentially more time or are too complex for classical computers to deal with. Today, researchers and engineers are building devices, that are capable of performing quantum computations, hence known as quantum computers. Algorithms, fundamentally different from classical ones, that run on a realistic model of quantum computation are referred to as quantum algorithms [1]. Many quantum algorithms have been proven to offer computational speed-ups over classical algorithms simply by design. Two well-known quantum algorithms are Shor's algorithm for factoring (superpolynomial[1] speed-up), and Grover's algorithm for searching in an unstructured database (quadratic

*Department of Electrical Engineering, akmandoi@iitk.ac.in
†Department of Electrical Engineering, aryab@iitk.ac.in
‡Department of Physics, mohantyj@iitk.ac.in

[1] The best known classical algorithm takes $2^{\tilde{O}(n^{1/3})}$ time, however Shor's algorithm does the same in $\tilde{O}(n^3)$ time.

speed-up [2]). In this report, we shall see how quantum algorithms are useful for speeding up the procedure of solving an SDP.

1.3. Preliminaries

Oracles are frequently used in both classical as well as quantum algorithms. An oracle can be conceived as a 'black-box' that gives an output corresponding to an input. The structure of an oracle depends on the algorithm where it is used. Classical oracles take numbers as inputs and return numbers as outputs, on the other hand, quantum oracles take input and return outputs both in the form of states.

The **input model** of an algorithm is also an important factor that affects its running time. It is not necessary that we always receive the inputs in terms of numbers or arrays. Input model where state vectors or density matrices are also acceptable have an inherent capacity to extract other physical information from the inputs. Quantum State Input model is one such example, and using this model can lead to speedups that may not be possible using any other model. Sparse Input model is one where we simply provide sparse matrices and vectors as inputs. We shall be primarily working with this model.

We shall say an algorithm has time complexity or (query complexity) T if it uses at most T elementary classical operations, quantum gates, and quantum queries for the input (wherever it is stored). Throughout this report, we shall use the following notation for time complexity:

$$\tilde{O}(f(a,b,c)) := O(f(a,b,c) \cdot polylog(f(a,b,c),d,e))$$

where $O(\cdot)$ is the big O notation. There are several parameters that need to be defined before we look at the algorithms in detail. $(C, A_1, \ldots, A_m)$, Hermitian matrices of dimension $n \times n$, and b, a m dimensional vector with real entries $(b_1, \ldots, b_m)$, constitute the input to the SDP. Further, s denotes the row-sparsity of a matrix (meaning that there are at most s non-zero elements per row), δ denotes the additive error allowed in the approximation of the objective. R is upper bound to the trace of the primal variable X, and r is the upper bound to the l_1 norm of the dual variable y.

2. Arora & Kale's SDP solver

Arora and Kale developed a general primal-dual approach to solve SDPs using the matrix multiplicative weights update rule. In contrast to other classical algorithms like [7] that have polynomial dependence on n, m and s, and polylogarithmic dependence on other parameters $(R, 1/\varepsilon)$ [3], this algorithm has only linear dependence on mn, and polynomial dependence on others.

Motivation
The SDP based approximation algorithms yield better approximation ratio than the LP based algorithms for NP-hard problems like MAXCUT, SPARSEST CUT, etc. [4]. For LPs, the primal-dual method and the rounding algorithm are two standard ways of using linear-programming relaxations to design approximation algorithms.

The primal-dual approximation algorithms for LPs incrementally build a dual solution together with and primal solution, updating them at each step using "combinatorial" methods [4]. In short we can say this update rule alternates between primal and dual guesses and quickly converges to the optimum.

Since SDPs also satisfy duality, this idea has been extended to develop primal-dual algorithms to solve SDPs.

The Algorithm
Let us consider the primal (1) and dual (2) SDP problems defined above. In this framework, the optimization problem will be changed into a feasibilty problem, where we make guesses for the optimum objective value of the SDP and progressively update our guesses using binary search. If α is our current guess, then we will add a constraint $b \cdot y \leq \alpha$ to the dual problem and check for feasibility. If it is feasible then the theoretical optimum value OPT of the objective must be less than or equal to α and we can decrease α to get closer to OPT; otherwise we increase the α.

Let's look at some definitions and results [4, 10] that are useful for the algorithm.

Definition ($\texttt{ORACLE}(X)$): *It is an oracle (1.3) that outputs a vector y from the polytope*

$$D_\alpha := \{y \in \mathbb{R}^m : \; b \cdot y \leq \alpha, \\ \sum_{j=1}^{m} (\mathrm{tr}(A_j X^{(t)})) y_j - \mathrm{tr}(C X^{(t)}) \geq 0, \\ y \geq 0\}$$

$$\sum_{j=1}^{m} \mathrm{tr}(A_j X^{(t)})) y_j - \mathrm{tr}(C X^{(t)}) = \sum_{j=1}^{m} \mathrm{tr}\left((A_j y_j - C) X^{(t)}\right)$$

Note that, the dual matrix constraint (2) $\sum_{j=1}^{m} (A_j) y_j - C \geq 0$ and the condition that $X^{(t)}$ is a PSD matrix gives the above constraint involving A_j and

[2] Grover search takes $\tilde{O}(\sqrt{n})$ time compared to $\tilde{O}(n)$ time of linear search.

[3] In particular it is claimed that the best classical algorithm has $O(m(m^2 + n^{2.373} + mns) \cdot polylog(R, m, n, 1/\varepsilon))$ complexity [7]

C is equivalent to the inner product of the dual matrix constraint (2) $\sum_{j=1}^{m} A_j y_j - C \geq 0$ and a PSD matrix $X^{(t)}$.

Now if we take only the specific candidate solution then we get the relaxed linear constraint in y:

$$\sum_{j=1}^{m} \operatorname{tr}(A_j X^{(t)}) y_j - \operatorname{tr}(C X^{(t)}) \geq 0 \tag{3}$$

We can see that this is a LP relaxation of the Linear Matrix Inequality constraint of the dual.

For simplicity we will assume is that $tr(X) \leq R$ in the primal problem, which is just a scaling constraint. This will automatically hold if we fix $A_1 = I$ and $b_1 = R$ in the primal problem 1.

The running time of the algorithm will also depend on the width parameter w of the $\texttt{ORACLE}(X)$.

Definition (Width parameter):

$$w := \max_{y \in D_\alpha} \left\| \sum_{j=1}^{m} y_j A_j - C \right\|_2$$

It can easily shown that $R + 1$ is an upper bound for w (using triangle inequality) [10]. As high width leads to slow progress, this ensures that the algorithm is fast.

Result 1 (Matrix Multiplicative Weights Update method): Let $M^{(t)}$ be such that $0 \leq M^{(t)} \leq I$ for every t. Fix $\varepsilon \leq 1/2$ and let $\varepsilon' = -ln(1-\varepsilon)$. Assign $W^{(t+1)} = e^{-\varepsilon' \sum_{\tau=1}^{t} M^{(\tau)}}$ and $X^{(t+1)} = \frac{RW^{(t+1)}}{tr(W^{(t+1)})}$. Then

$$\sum_{j=1}^{T} \operatorname{tr}(M^{(t)} X^{(t)}) \leq (1+\varepsilon)\lambda_{min} \left(\sum_{j=1}^{T} M^{(t)} \right) + \frac{ln(n)}{\varepsilon}$$

Result 2 (Dual feasibility): Suppose $\texttt{ORACLE}(X)$ never fails for $T = \frac{16R^4 ln(n)}{\delta^2 \alpha^2}$ iterations. Then $\bar{y} = \frac{\delta\alpha}{R} e_1 + \frac{1}{T} \sum_{t=1}^{T} y^{(t)}$ where $e_1 = (1, 0, 0, \ldots, 0, 0)$ is dual feasible with objective value atmost $(1+\delta)\alpha$. [4]

Here we have the full algorithm.

```
Algorithm 1: Arora-Kale Primal-Dual Algorithm
Input: Given parameters $n, m, R, \delta, \alpha, w$
Result: Get a dual feasible $y^*$ with objective value at
        most $(1+\delta)\alpha$
$\varepsilon \leftarrow \frac{\delta\alpha}{2wR}$, $\varepsilon' \leftarrow -ln(1-\varepsilon)$, and $T \leftarrow \frac{8w^2R^2ln(n)}{\delta^2\alpha^2}$
while $t \leq T$ do
    if ORACLE($X$) fails with input $X^{(t)}$ then
        return $X^{(t)}$
    else
        ORACLE($X$) outputs $y^{(t)}$
        $M^{(t)} = \frac{\sum_{j=1}^{m} A_j y_j^{(t)} - C + wI}{2w}$
        $W^{(t+1)} = e^{-\varepsilon' \sum_{\tau=1}^{t} M^{(\tau)}}$
        $X^{(t+1)} = \frac{RW^{(t+1)}}{\operatorname{tr}(W^{(t+1)})}$
    end
end
```

Analysis of the Algorithm Going through the steps of the algorithm we see that:

- We can see that this algorithm runs for $T = O\left(\frac{log(n)}{\delta^2}\right)$ iterations which according to **Result 2**, will give a dual feasible solution which has value less than $(1+\delta)\alpha$.
- If the $\texttt{ORACLE}(X)$ succeeds in finding a y, then we say $X^{(t)}$ is either primal infeasible or the objective value $\operatorname{tr}(CX^{(t)}) \leq \alpha$. We shall prove this using contradiction. Let the opposite be true, i.e., $\texttt{ORACLE}(X)$ succeeds and $\operatorname{tr}(CX^{(t)}) \geq \alpha$.
$$\sum_{j=1}^{m} \operatorname{tr}(A_j X^{(t)}) y_j - \operatorname{tr}(CX^{(t)}) \leq \sum_{j=1}^{m} b_j y_j - tr(CX^{(t)}) < 0$$
The first inequality comes from the primal constraint, and the second one comes from our above assumption. However, this contradicts (3). When the $\texttt{ORACLE}(X)$ doesn't succeed, the corresponding $X^{(t)}$ (that is simply returned) must be primal feasible as the value $tr(CX^{(t)}) \geq \alpha$.
- The matrix multiplicative weights update rule has the form $x^{(t+1)} \propto x^{(t)} e^{\varepsilon f(x)}$, where $f(x)$ is a feedback function which are found by combinatorial algorithms [4]. Here the feedback function is $M^{(t)} = \frac{\sum_{j=1}^{m} A_j y_j - C + wI}{2w}$. Note that for **Result 1** to be applicable here $M^{(t)}$ should satisfy $0 \leq M^{(t)} \leq I$. Here $\lambda_{max}(M^{(t)}) = \frac{\lambda_{max}(\sum_{j=1}^{m} A_j y_j - C) + w}{2w} \leq 1$ [5].
- In the case when we have the output $y^{(t)}$ from the $\texttt{ORACLE}(X)$, the steps involving updation of $M^{(t)}$ and $W^{(t)}$ represent the matrix multiplicative weights update rule. Matrix exponentials play a vital role in the algorithm and to make exponentiation faster the

[4] δ is the additive error allowed in the optimum value

[5] λ_{max} denotes maximum eigenvalue.

authors use efficient algorithms that compute them approximately(Section 5 of [4]).

- In the last step, we update our primal candidate solution, which has a trace R, with a normalized $W^{(t)}$.

We found that after T iterations the SDP converges. In each iteration, the two most expensive steps are the implementation of $\texttt{ORACLE}(X)$ and computation of the matrix exponential. Taking that into account we can the complexity of the general classical SDP solver to be $\tilde{O}(nms(\frac{Rr}{\varepsilon})^4 + ns(\frac{Rr}{\varepsilon})^7)$, where R and r are constraints corresponding to primal and dual solutions ($tr(X) \leq R$ (for primal) and $||y||_1 \leq r$ (for dual)).

3. Brandão & Svore's quantum speed-up

In 2016, Brandão and Svore gave the first quantum algorithm for solving SDPs offering a speed-up over any classical algorithm [10]. The key observation that Brandão and Svore made in the Arora-Kale framework was that the matrix

$$\rho := \frac{e^{-\eta H}}{\mathrm{tr}(e^{-\eta H})}$$

can be interpreted as a $log(n)$-qubit quantum state called Gibbs state, and H as a Hamiltonian. This allowed them to introduce quantum subroutines into the algorithm that are responsible for a speed-up compared to Arora and Kale's algorithm.

The main ideas behind their contributions can be summarized as follows [10]:

1. It can be shown that the Gibbs sampler operation (3) can replace the the procedure of how $\texttt{ORACLE}(X)$ (defined in the previous section) is queried for updating the primal candidates $X^{(t)}$.
2. The Gibbs state preparation (done using the Gibbs sampler operation) procedure must be based on amplitude amplification[6] [2]. Hence, the Gibbs state can be efficiently prepared as a quantum state using a Gibbs sampler on a quantum computer.
3. The quantum algorithm for preparing Gibbs states have linear dependence on sparsity of the Hamiltonians. This sparsity in turn also depends on m. It can be shown that the Hamiltonians can be sparsified by random sampling, so that their sparsity only depends on the original sparsity s of the input matrices.

Using the above ideas, Brandão and Svore gave the quantum SDP-solver algorithm for a *special case* of SDP, where $b_j \geq 1 \ \forall j \in [m]$ and $\alpha \geq 1$ [7]. Now, we focus on the core quantum part of Brandão and Svore's algorithm.

In the algorithm, we shall need access to the three oracles $\mathcal{O}_{index}$, $\mathcal{O}_A$ and $\mathcal{O}_b$ defined below.

$$\begin{aligned} \mathcal{O}_{index}(j,k,l) &:= \text{column index of the } l^{th} \text{ non-zero} \\ &\quad\ \text{entry in the } k^{th} \text{ row of } A_j \\ \mathcal{O}_A(j,k,l) &:= (A_j)_{k,\mathbf{index}(j,k,l)} \\ \mathcal{O}_b(j) &:= b_j \end{aligned}$$

The first step of the algorithm, i.e., reducing the optimization problem to a feasibility problem is similar to that of Arora-Kale's framework. In other words, if α is our guess for the optimal objective value, it will be iteratively updated by binary search depending on whether the dual feasible vector y corresponds to a feasible solution.

Definition (Gibbs sampler) *[Definition 1 of [10]]: Given H a Hamiltonian and $\mathcal{O}[H]$ an oracle that can query its entries, $\texttt{GibbsSampler}(\mathcal{O}[H], \nu)$ is a quantum operation which has access to the oracle $\mathcal{O}[H]$ and finally outputs a state ρ such that $\left|\left|\rho - \frac{e^H}{\mathrm{tr}(e^H)}\right|\right|_1 \leq \nu$.*

Given a candidate dual feasible vector $y^{(t)}$, Gibbs sampler prepares the new primal candidate $X^{(t+1)} \propto e^{H^{(t)}}$ as a $log(n)$-qubit quantum state $\rho^{(t)} := \frac{X^{(t)}}{\mathrm{tr}(X^{(t)})}$ in much less time than what is needed to compute $X^{(t)}$ as an $n \times n$ matrix [12].

Next step is to implement the oracle $\texttt{ORACLE}(X)$ (exactly the same as in Arora-Kale framework) which can be efficiently done using sufficient number of copies of $\rho(t)$. Further, these copies are used to estimate $\mathrm{tr}(A_j\rho^{(t)})$ and $\mathrm{tr}(A_jX^{(t)})$. In quantum physics $\mathrm{tr}(A\rho)$ is simply called the expectation value of A for a quantum system in a thermal state corresponding to H. This step is based on Jaynes's principle of maximum entropy [8].

Brandão and Svore's algorithm hence leads to an ε-approximate quantum SDP solver with an overall complexity [13]:

$$\tilde{O}\left(\sqrt{mn}s^2\left(\frac{Rr}{\delta}\right)^{32}\right) \tag{4}$$

As mentioned before, Brandão and Svore's algorithm works for the special form of SDP with $b_i \geq 1 \ \forall i \in [m]$ and $\alpha \geq 1$. The authors describe how a general SDP of the form 1 can be transformed into this special case so that the algorithm is applicable. Additionally, they also show that arbitrary SDPs can be solved in roughly the same time as that of these special SDPs. However, the transformation significantly worsens the dependence of the complexity on the parameters R, r, and δ. Therefore,

[6] Amplitude amplification is a subroutine of the Grover's algorithm, that is responsible for its quadratic speedup.

[7] α, as described later, denotes our guess for the optimal value of the objective.

this algorithm may not work well in the regimes where the complexity dependence on R, r, s, and $1/\delta$ are comparable to that of m or n. Another potential drawback that comes with this approach is the case when $\alpha < 1$. Given the SDP of the form (2) with b_i's greater than 1, and y_i's are non-negative, $1 > \alpha > 0$ is a possible case for the SDP which is not directly tackled. Instead the dual objective is modified by rescaling the b_i's by $1/\alpha$. This new SDP now satisfies Brandão and Svore's requirement of the special case SDP, but now the error δ is redefined to be δ/α. As α approaches 0, the complexity (4) increases.

Along with achieving a better complexity (w.r.t. the number of dimensions and constraints, of the SDP), Brandão and Svore also proved lower bounds on the complexity of solving SDPs using classical and quantum algorithms in terms of n and m keeping other parameters as constants. They showed that at least $\Omega(n+m)$ calls to a classical oracle and $\Omega(\sqrt{n}+\sqrt{m})$ calls to a quantum oracle are necessary to solve a SDP[8]. This helped them to prove that their algorithm (although not optimal) is close to the theoretically best possible algorithm in terms of complexity and motivated to continue searching for an optimal algorithm in terms of n and m. In fact, as we will see in the next section, recently, there have been further improvements in their algorithm that resulted in the optimal dependence on n and m.

Recently, they also gave an improved SDP-solver for the quantum state input model [?], and applied the same to the problem of *shadow tomography*, giving the first non-trivial application of a quantum SDP-solver [12]. We shall briefly see the idea behind this and other applications in a later section.

4. Further Refinements

In 2019, Brandão, Li, Svore et al. [13] published two new quantum algorithms for solving SDPs, which provided better quantum speed-ups. The first algorithm uses a sparse input model similar to the above algorithm, and had a run time of $\tilde{O}\left(s^2\left(\frac{\sqrt{m}}{\varepsilon^{10}}+\frac{\sqrt{n}}{\varepsilon^{12}}\right)\right)$, compared to the $\tilde{O}\left(\sqrt{mn}\right)$ dependence of the above algorithm. This algorithm has optimal dependence on m and n, matching the dependence of the lower bound.

The main ideas that were different here as compared to the above algorithm are the introduction of the zero-sum game approach for Matrix Multiplicative Update Method (MMUW), and the Fast Quantum OR lemma[13]. They used a different feasibility problem and introduced a new oracle $\mathcal{O}_V$ based on it, which helps to view the SDP feasibility problem as a game. They used MMUW to approximate the feasible solutions with $\tilde{O}\left(\frac{log(n)}{\varepsilon^2}\right)$ queries to the oracle. This was similar to Arora-Kale's framework.

The Fast Quantum OR lemma [9, 5] is used to implement the $\mathcal{O}_V$ that verifies feasibility. As we saw above, to implement the oracle we need to generate multiple copies of $\rho(t)$ so that we can search among the m constraints to detect a single violation of the oracle. The cost of generating $\rho(t)$ is $O(\sqrt{n})$ (ignoring other parameters) and the cost of searching among the m constraints is $O(\sqrt{m})$ (using Grover's search)[13]. Therefore the overall cost to implement the oracle of the previous algorithms was $O(\sqrt{mn})$. But here the use of Fast Quantum OR lemma helps us to detect a single violation with only a single copy of $\rho(t)$. This allows us decouple the complexity dependence on m and n in the oracle implementation to $O(\sqrt{m}+\sqrt{n})$.

The second algorithm includes quantum state input model, in which the matrices that are given as input, are quantum states. The input matrix A_j can be decomposed into the basis density operators of the Hilbert space. They used a oracle $\mathcal{O}_\mu$ to write the density operators, that are given in the decomposed form of the input matrices A_j, using pure states[9]. The run time of this input model was found to be $\tilde{O}(\sqrt{m}+poly(r))\cdot poly(log\, m, log\, n, B, \varepsilon^{-1})$, where B refers to the upper bound of the trace norm of the input matrices A_j, which will be equivalent to the sum of the modulus of coefficients of density operators in the decomposition of A_j. It is important to note that this is a non-standard input model and is thus incomparable to the normal matrix input model. This is useful only in certain applications of SDP like in Quantum Information, where the input matrices are inherently quantum states or measurements. In the nest section, we will see this model being applied to the problem of shadow tomography.

5. Applications

5.1. Shadow Tomography using SDP solvers

Taking measurement on a quantum state, immediately destroys the quantum state, that too resulting in probabilistic outcomes. So creating another state that closely resembles the initial state is a challenging task.

In shadow tomography, one does not measure the entire quantum state, rather one measures certain linear functions, such as quantum fidelity, entanglement witness, etc. The full classical descriptions of quantum systems may be excessive for many concrete tasks. These target linear function measurements would be sufficient to accurately

[8] classical lower bound: $\Omega(n+m)$,
quantum lower bound: $\Omega(\sqrt{n}+\sqrt{m})$

[9] Pure state: State which cannot be represented as a convex combination of other basis states

predict certain properties of the quantum system.

We would find a state for which $\mathrm{Tr}[\rho E_i]$ is close to the actual expectation value of E_i up to an additive error ε, rather than finding the expectation values for each E_is.

Using Jaynes Principle of maximum entropy[6], the general form of solution of the given problem shows that there exist a state:

$$\rho := \frac{e^{\sum_i \lambda_i E_i}}{\mathrm{tr}(e^{\sum_i \lambda_i E_i})}$$

which has the same expectation values on the E_is as the original state ρ, where $\lambda_i \in \mathbb{R}$. Therefore one can solve the problem by finding the correct λ_is.

We would use the idea of SDP feasibility problem to search for a state σ, for which the expectation values are ε close to actual expectation. The entire formalism is applicable for low rank measurements E_is.

Now consider two oracles:
Oracle 1 for trace of E_i : O_{Tr} unitary operator: $O_{Tr} |i\rangle |0\rangle = |i\rangle |Tr[E_i]\rangle$
Oracle 2 for preparing E_i : O unitary such that $O |i\rangle \langle i| \otimes |0\rangle \langle 0| O^\dagger = |i\rangle \langle i| \otimes |\psi_i\rangle \langle \psi_i|$, where $|\psi_i\rangle \langle \psi_i|$ represents the purified state of the Gibbs state.
We would implement these oracles to show the complexity of this protocol.
Let the E_is be decomposed as $V_i P_i V_i^\dagger$, where projectors P_i are given by $\sum_{i=1}^{r_i} |i\rangle \langle i|$.
Then for oracle 1, we just need to output r_i, as $\mathrm{Tr}[E_i]$ = $\mathrm{Tr}[P_i]$. Oracle 2 can then be implemented efficiently by creating a maximally entangled state between the subspace spanned by P_i and a purification and applying V_i to one half of it. Purification for the Gibbs state is given by:

$$|\psi_i\rangle := \frac{1}{\sqrt{r_i}} \sum_{i=1}^{r_i} (V_i \otimes I) |i, i\rangle \tag{5}$$

The main idea to define the above oracles is that, SDP solver will maintain and update a description of $\bar{\rho}$ per iteration. Our aim is to know whether $\mathrm{Tr}(E_i\bar{\rho}) \equiv \mathrm{Tr}(E_i\rho)$ for all i $\in [1, m]$ or whether there is a discrepancy. We design for each 'i' a projection for the following procedure: We perform multiple independent SWAP tests between Ei/Tr[Ei] (from Oracle 2) and ρ, $\bar{\rho}$ respectively and then, accept when the statistics of both SWAP tests (one for ρ, the other for $\bar{\rho}$) are close.
Hence, one can apply our fast quantum OR lemma [9, 5] on these projections to find if such i^* it exists.
This SDP solver will give the $\bar{\rho}$ using atmost poly(log m, log n, r, ε^{-1}) samples of ρ, and atmost $\sqrt{m}$poly(log m, log n, r, ε^{-1}) queries to oracles.
As we are dealing with small rank r, say r = O(poly log n), the gate complexity of the entire procedure becomes $\tilde{O}(\sqrt{m} polylog(n))$.

In [12], more sophisticated techniques were used to obtain complexity of $\tilde{O}(log(n) \cdot log^4 m/\varepsilon^4)$, which improved the complexity of shadow tomography, as previously done by Aaronson[11].

5.2. Quantum Error Recovery

In quantum communication, one has to deal efficiently with the error cumulated in the quantum channel. One uses techniques of Quantum error correction so as to remove errors. Quantum errors can arise in quantum channels in the form of bit flips, change in phase of the quantum system, etc. The previous quantum error correction codes used classical correction methods in quantum theories, which relies on the fact on correction on the subset of errors, however it does not depend on the input state. For quantum error recovery, we will maximize the entanglement fidelity, thus maximizing the closeness of input and output states, and show that the problem would come out to be a SDP.
Any quantum system can be completely described by the density operator ρ. We can denote the noise from the quantum channel by $\mathcal{E}$ and the recovery operation on the noisy quantum channel by $\mathcal{R}$. So the output of the channel would be $\mathcal{R}(\mathcal{E}(\rho))$.
The optimization problem in terms of fidelity operator can be written as:

$$\mathcal{R}^* = \arg\ \max_{\mathcal{R}} F_o(\mathcal{R} o \mathcal{E}) \tag{6}$$

The maximization is over all possible recovery operations. Using entanglement fidelity as the Fidelity operator, we would show how the maximization problem is a SDP.
Lets say the mixed quantum state ρ lies in the Hilbert space denoted by $\mathcal{L}(\mathcal{H})$. It can converted into a pure state, by working on an auxiliary space A. The pure state is given by:

$$\rho = tr_{\mathcal{A}} |AH\rangle \langle AH| \tag{7}$$

where the trace is taken over the space A.
Entanglement fidelity is then given by:

$$F(\rho, \mathcal{B}) = \langle AH| |I_{\mathcal{A}} \otimes \mathcal{B}(|AH\rangle \langle AH|) |AH\rangle \tag{8}$$

which shows how well $\mathcal{B}$, which is a quantum channel, preserves the entanglement.
When we know B_is for $\mathcal{B}$, the above expression reduces to a simpler version:

$$F(\rho, \mathcal{B}) = \sum_i |Tr(\rho B_i)|^2 \tag{9}$$

If we define $X_{\mathcal{E}} = \sum_k |E_k\rangle \langle E_k|$, and using the known relations which are derived from a system's state in a coupled Hilbert space $H_1 \otimes H_2$, Entanglement fidelity is given by:

$$F(\rho, \mathcal{R} o \mathcal{E}) = tr(X_{\mathcal{R}} C_{\rho,\mathcal{E}}) \tag{10}$$

So by maximizing this entanglement fidelity, the optimization problem can then be written in the form given by:

$$X^*_{\mathcal{R}_\rho} = argmax_X tr(X C_{\rho,\mathcal{E}})) \tag{11}$$

such that $\quad X\geq 0, \quad tr_{\mathcal{H}}(X) = 1$
Here $C_{\rho,\mathcal{E}}$, represents a matrix that depends on the noise $\mathcal{E}$ as well as the input quantum state ρ, given by:

$$C_{\rho,\mathcal{E}} = \sum_i |\rho E_i^\dagger\rangle \langle \rho E_i^\dagger| \tag{12}$$

Equation 11 is a SDP, which can be solved efficiently using quantum speed ups which are discussed above.

6. Conclusion and Future Work

Algorithm	Query Complexity
Arora Kale Algorithm	$\tilde{O}\left(nms(\frac{Rr}{\varepsilon})^4 + ns\left(\frac{Rr}{\varepsilon}\right)^7\right)$
Brandão Svore Algorithm	$\tilde{O}\left(\sqrt{mn}s^2\left(\frac{Rr}{\delta}\right)^{32}\right)$
Further Improvements	$\tilde{O}\left(s^2\left(\frac{\sqrt{m}}{\varepsilon^{10}} + \frac{\sqrt{n}}{\varepsilon^{12}}\right)\right)$

Table-1: Query complexities of different algorithms

It is clear that optimal dependence have been achieved in the parameters m and n but still speed-ups can be made to improve the dependence of other parameters.

We also saw how these quantum SDP solvers can be used in some applications such as Shadow tomography and Quantum error recovery, which improved the efficiency of the previous algorithms for the problems.

Although there are tools offered by software toolkits such as Qiskit(IBM), Ocean(D-Wave) among others, that are capable of solving Quadratic Programs (QPs) using quantum algorithms, we currently don't know of any tools that allow us to solve SDPs using quantum algorithms, we hope to work such implementation aspects.

We would also like to have these quantum SDP solvers to be robust to noise in the current noisy intermediate-scale quantum era (NISQ) era.

References

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[2] G. Bassard, P. Hoyer, and M. Mosca. *Quantum amplitude amplification and estimation.* 2002.

[3] S. Arora, E. Hazan, and S. Kale. *Fast algorithms for approximate semidefinite programming using the multiplicative weights update method.* 2005.

[4] S. Arora and S. Kale. *A Combinatorial, Primal-Dual Approach to Semidefinite Programs.* 2007.

[5] Pawel Wocjan Daniel Nagaj and Yong Zhang. *Fast amplification of QMA.* 2009.

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

[7] A. Sidford Y. T. Lee and S. C. Wong. "A faster cutting plane method and its implications for combinatorial and convex optimization". In: *Proceedings of 56th IEEE FOCS.* 2015, pp. 1049–1065.

[8] J. V. Apeldoorn et al. "Quantum SDP-solvers: Better upper and lower bounds". In: *Proceedings of 58th IEEE FOCS.* 2017.

[9] Cedric Yen-Yu Lin Aram W. Harrow and Ashley Montanaro. "Sequential measurements, disturbance and property testing." In: *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms.* 2017.

[10] F. Brandao and K. Svore. *Quantum Speed-ups for Semidefinite Programming.* 2017.

[11] Scott Aaronson. *Shadow Tomography of Quantum States.* 2018.

[12] J. V. Apeldoorn and A. Gilyen. *Improvements in Quantum SDP-Solving with Applications.* 2019.

[13] F. Brandão et al. *Quantum SDP Solvers: Large Speed-ups, Optimality, and Applications to Quantum Learning.* 2019.