Quantum Speed-ups for Semidefinite Programming

Fernando G.S.L. Brandão
Caltech

Krysta Svore
Microsoft Research

QIP 2017
Quantum Algorithms

Exponential speed-ups:
Simulate quantum physics, factor big numbers (Shor’s algorithm), ...

Polynomial Speed-ups:
Searching (Grover’s algorithm), ...

Heuristics:
Quantum annealing, adiabatic optimization, ...
Quantum Algorithms

Exponential speed-ups:
Simulate quantum physics, factor big numbers (Shor’s algorithm), ...,

Polynomial Speed-ups:
Searching (Grover’s algorithm), ...

Heuristics:
Quantum annealing, adiabatic optimization, ...

This Talk:
Solving Semidefinite Programming belongs here
Semidefinite Programming

... is an important class of convex optimization problems

\[
\begin{align*}
\max & \quad \text{tr}(CX) \\
\forall j \in [m], \quad & \text{tr}(A_j X) \leq b_j \\
X & \geq 0.
\end{align*}
\]

Input: \( n \times n \), \( s \)-sparse matrices \( C, A_1, \ldots, A_m \) and numbers \( b_1, \ldots, b_m \)

Output: \( X \)
Semidefinite Programming

... is an important class of convex optimization problems

\[
\begin{align*}
\text{max } \text{tr}(CX) \\
\forall j \in [m], \quad \text{tr}(A_j X) & \leq b_j \\
X & \geq 0.
\end{align*}
\]

Input: \(n \times n\), \(s\)-sparse matrices \(C, A_1, ..., A_m\) and numbers \(b_1, ..., b_m\)

Output: \(X\)

Linear Programming: special case

Many applications (combinatorial optimization, operational research, ....)

Natural in quantum (density matrices, ...)
Semidefinite Programming

... is an important class of convex optimization problems

$$\begin{align*}
\max & \quad \text{tr}(CX) \\
\text{subject to} & \quad \forall j \in [m], \quad \text{tr}(A_j X) \leq b_j \\
& \quad X \geq 0.
\end{align*}$$

Input: \(n \times n\), \(s\)-sparse matrices \(C, A_1, \ldots, A_m\) and numbers \(b_1, \ldots, b_m\)

Output: \(X\)

Linear Programming: special case

Many applications (combinatorial optimization, operational research, ....)

Natural in quantum (density matrices, ...)

Algorithms

Interior points: \(O((m^2ns + mn^2) \log(1/\delta))\)

Multiplicativc Weights: \(O((mns (\omega R)/\delta^2))\)
Semidefinite Programming

... is an important class of convex optimization problems

\[
\begin{align*}
\max & \quad \text{tr}(CX) \\
\forall j \in [m], & \quad \text{tr}(A_j X) \leq b_j
\end{align*}
\]

Input: \(n\times n\) matrices \(C, A_1, \ldots, A_m\) and numbers \(b_1, \ldots, b_m\)

Output: \(X\)

Linear Programming: special case

Many applications (combinatorial optimization, operational research, ....)

Natural in quantum \((\text{density matrices})\)

Algorithms

- Interior points: \(O((m^2ns + mn^2)\log(1/\delta))\)
- Multiplicative Weights: \(O((mns (\omega R)/\delta^2))\)

Are there quantum speed-ups for SDPs/LPs?

Natural question. But unexplored so far
SDP Duality

**Primal:**
\[
\forall j \in [m], \quad \operatorname{tr}(A_j X) \leq b_j \quad \text{Opt}_{\text{primal}} = \text{Opt}_{\text{dual}}
\]

**Dual:**
\[
\min b.y \\
\sum_{j=1}^{m} y_j A_j \geq C \\
y \geq 0.
\]

\[y: m\text{-dimensional vector}\]

Under mild conditions: \(\text{Opt}_{\text{primal}} = \text{Opt}_{\text{dual}}\)
Size of Solutions

Primal: \[ \forall j \in [m], \quad \operatorname{tr}(A_j X) \leq b_j \]
\[ X \geq 0. \]

R parameter: \[ \operatorname{Tr}(X_{\text{opt}}) \leq R \]

Dual: \[ \min b.y \]
\[ \sum_{j=1}^{m} y_j A_j \geq C \]
\[ y \geq 0. \]

r parameter: \[ \sum_i (y_{\text{opt}})_i \leq r \]
SDP Lower Bounds

Even to write down optimal solutions take time:

**Primal** \((n \times n) \text{ PSD matrix } X\): \(\Omega(n^2)\)

**Dual** \((m \text{ dim vector } y)\): \(\Omega(m)\)
SDP Lower Bounds

Even to write down optimal solutions take time:

**Primal** \((n \times n\) PSD matrix \(X\)): \(\Omega(n^2)\)
**Dual** \((m\) dim vector \(y\)): \(\Omega(m)\)

Even just to compute optimal value requires:

**Classical**: \(\Omega(n+m)\) (for constant \(r, R, s, \delta\))
**Quantum**: \(\Omega(n^{1/2} + m^{1/2})\) (for constant \(r, R, s, \delta\))

Easy reduction to search problem

(Appeldoorn, Gilyen, Gribling, de Wolf)

**Quantum**: \(\Omega(nm)\) if \(n \cong m\)
\(\min(m, n) (\max(m, n))^{1/2}\)

See poster this afternoon

(R, s, \(\delta = O(1)\) but not \(r\))
(R, s, \(\delta = O(1)\) but not \(r\))
Quantum Algorithm for SDP

**Result 1:** There is a quantum algorithm for solving SDPs running in time $n^{1/2} m^{1/2} s^2 \text{poly}(\log(n, m), R, r, \delta)$

**Input:** $n \times n$, s-sparse matrices $C, A_1, ..., A_m$ and numbers $b_1, ..., b_m$
Quantum Algorithm for SDP

**Result 1:** There is a quantum algorithm for solving SDPs running in time $n^{1/2} m^{1/2} s^2 \text{poly}(\log(n, m), R, r, \delta)$

**Input:** $n \times n$, $s$-sparse matrices $C, A_1, \ldots, A_m$ and numbers $b_1, \ldots, b_m$

**Normalization:** $||A_i||, ||C|| \leq 1$

**Output:** Samples from $y/||y||_1$ and value $||y||_1$ and/or Quantum Samples from $X/\text{tr}(X)$ and value $\text{tr}(X)$

Value $\text{opt} \pm \delta$

(output form similar to HHL Q. Algorithm for linear equations)
Quantum Algorithm for SDP

**Result 1:** There is a quantum algorithm for solving SDPs running in time $n^{1/2} m^{1/2} s^2 \text{poly}(\log(n, m), R, r, \delta)$

**Oracle Model:** We assume there’s an oracle that outputs a chosen non-zero entry of $C, A_1, \ldots, A_m$ at unit cost:

$$|j, k, l, z\rangle \rightarrow |j, k, l, z \oplus (A_j)_{k,f_{j,k}(l)}\rangle$$

$$f_{j,k} : [r] \rightarrow [n]$$

- choice of $A_j$
- row $k$
- $l$ non-zero element
Quantum Algorithm for SDP

Result 1: There is a quantum algorithm for solving SDPs running in time $n^{1/2} m^{1/2} s^2 \text{poly}(\log(n, m), R, r, \delta)$

The good:
Unconditional Quadratic speed-ups in terms of $n$ and $m$

Close to optimal: $\Omega(n^{1/2} + m^{1/2})$ lower bound
Quantum Algorithm for SDP

**Result 1:** There is a quantum algorithm for solving SDPs running in time $n^{1/2} m^{1/2} s^2 \text{poly}(\log(n, m), R, r, \delta)$

**The good:**
Unconditional Quadratic speed-ups in terms of $n$ and $m$

Close to optimal: $\Omega(n^{1/2} + m^{1/2})$ q. lower bound

**The bad:**
Terrible dependence on other parameters: $\text{poly}(\log(n, m), R, r, \delta) \leq (Rr)^{32} \delta^{-18}$

Close to optimal: no general super-polynomial speed-ups
**Quantum Algorithm for SDP**

**Result 1:** There is a quantum algorithm for solving SDPs running in time $n^{1/2} m^{1/2} s^2 \text{poly}(\log(n, m), R, r, \delta)$

**Special case:**
If the SDP is s.t. $b_i \geq 1$ for all $i$, there is no dependence on $r$ (size of dual solution)
Result 2: There is a quantum algorithm for solving SDPs running in time $T_{\text{Gibbs}} m^{1/2} \text{poly}(\log(n, m), s, R, r, \delta)$.
**Larger Speed-ups?**

**Result 2:** There is a quantum algorithm for solving SDPs running in time $T_{\text{Gibbs}} m^{1/2} \text{poly}(\log(n, m), s, R, r, \delta)$

**$T_{\text{Gibbs}} :=**$ Time to prepare on quantum computer Gibbs states of the form

$$\exp \left( \sum_{i=1}^{m} \nu_i A_i + \nu_0 C \right) / \text{tr}(\ldots)$$

for real numbers $|\nu_i| \leq O(\log(n), \text{poly}(1/\delta))$
**Larger Speed-ups?**

**Result 2:** There is a quantum algorithm for solving SDPs running in time $T_{\text{Gibbs}} m^{1/2}\text{poly}(\log(n, m), s, R, r, \delta)$

$T_{\text{Gibbs}} :=$ Time to prepare on quantum computer Gibbs states of the form

$$\exp \left( \sum_{i=1}^{m} \nu_i A_i + \nu_0 C \right) / \text{tr}(\ldots)$$

for real numbers $|\nu_i| \leq O(\log(n), \text{poly}(1/\delta))$

Can use **Quantum Gibbs Sampling** (e.g. Quantum Metropolis) as heuristic. Exponential Speed-up if thermalization is quick (poly #qubits = polylog(n))

Gives application of quantum Gibbs sampling outside simulating physical systems
### Result 3: There is a quantum algorithm for solving SDPs running in time $m^{1/2}\text{poly}(\log(n, m), s, R, r, \delta, \text{rank})$ with data in quantum form.

#### Quantum Oracle Model: There is an oracle that given $i$, outputs the eigenvalues of $A_i$ and its eigenvectors as quantum states

$$\text{rank} := \max (\max_i \text{rank}(A_i), \text{rank}(C))$$
Larger Speed-ups with “quantum data”

**Result 3:** There is a quantum algorithm for solving SDPs running in time $m^{1/2}\text{poly}(\log(n, m), s, R, r, \delta, \text{rank})$ with data in quantum form.

**Quantum Oracle Model:** There is an oracle that given $i$, outputs the eigenvalues of $A_i$ and its eigenvectors as quantum states.

**rank** := $\max (\max_i \text{rank}(A_i), \text{rank}(C))$

**Idea:** in this case one can easily perform the Gibbs sampling in $\text{poly}(\log(n), \text{rank})$ time.

**Limitation:** Not clear the relevance of the model. How to compare with classical methods in a meaningful way?
Special Case: Max Eigenvalue

Computing the max eigenvalue of $C$ is a SDP

$$\max \operatorname{tr}(CX): \quad \operatorname{tr}(X) = 1, \quad X \succeq 0$$
Computing the max eigenvalue of $C$ is a SDP

$$\text{max } \text{tr}(C X) : \quad \text{tr}(X) = 1, \quad X \geq 0$$

This is a well studied problem:

**Quantum Annealing** (cool down $-C$):

If we can prepare $e^{\beta C} / \text{tr}(e^{\beta C})$ for $\beta = O(\log(n)/\delta)$ can compute max eigenvalue to error $\delta$
Special Case: Max Eigenvalue

(Poulin, Wocjan ‘09) Can prepare $e^{\beta C}/\text{tr}(e^{\beta C})$ for s-sparse $C$ in time $\tilde{O}(s \sqrt{n})$ on quantum computer

Idea: Phase estimation + Amplitude amplification

$$C|\psi_i\rangle = E_i|\psi_i\rangle$$

$$\sum_i |\psi_i\rangle|\psi_i^*\rangle \rightarrow \sum_i |\psi_i\rangle|\psi_i^*\rangle|E_i\rangle \rightarrow \sum_i e^{-E_i/2}|\psi_i\rangle|\psi_i^*\rangle|E_i\rangle|0\rangle + \ldots$$

phase estimation

Post-selecting on “0” gives a purification of Gibbs state with $\text{Pr} > O(1/n)$

Using amplitude amplification can boost $\text{Pr} > 1-o(1)$ with $O(n^{1/2})$ iterations
General Case: Quantizing Arora-Kale Algorithm

The quantum algorithm is based on a classical algorithm for SDP due to Arora and Kale (2007) based on the multiplicative weight method. Let’s review their method

Assumptions:

We assume $b_i \geq 1$.
Can reduce general case to this with blow up of poly(r) in complexity

We also assume w.l.o.g. $A_1 = I, b_1 = R$
The Oracle

The Arora-Kale algorithm has an auxiliary algorithm (the ORACLE) which solves a simple linear programming:

\[
\text{ORACLE}(\rho)
\]

Searches for a vector \( y \) s.t.

i) \( y \in D_\alpha := \{ y : y \geq 0, \ b.y \leq \alpha \} \)

ii) \( \sum_{j=1}^{m} \text{tr}(A_j \rho) y_j - \text{tr}(C \rho) \geq 0 \)
Arora-Kale Algorithm

\[
\rho^1 = I/n, \quad \varepsilon = \frac{\delta}{2R}, \quad \varepsilon' = -\ln(1 - \varepsilon), \quad T = \frac{8R^2 \ln(n)}{\delta^2}
\]

For \( t = 1, \ldots, T \)

1. \( y^t \leftarrow \text{ORACLE}(\rho^t) \)

2. \( M^t = \left( \sum_{j=1}^{m} y_j^t A_j - C + RI \right) / 2R \)

3. \( W^{t+1} = \exp \left( -\varepsilon' \left( \sum_{\tau=1}^{t} M^\tau \right) \right) \)

4. \( \rho^{t+1} = W^{t+1} / \text{tr}(W^{t+1}) \)

Output: \( \bar{y} = \frac{\delta \alpha}{R} e_1 + \frac{1}{T} \sum_{t=1}^{T} y^t \) \quad \quad e_1 = (1, 0, \ldots, 0)
\[
\rho^1 = I/n, \quad \varepsilon = \frac{\delta}{2R}, \quad \varepsilon' = -\ln(1 - \varepsilon), \quad \gamma = \frac{8R^2 \ln(n)}{\delta^2}
\]

For \( t = 1, \ldots, T \)

1. \( y_t^* = \text{Oracle}(\rho_t) \)

2. \( W_t = \frac{1}{\gamma} \sum_{\tau=1}^{T} \left( \frac{2}{\gamma} \right)^{\tau-1} y_{\tau}^* \)

3. \( W_{t+1} = W_t + \rho_t \sum_{\tau=1}^{T} (2(2\tau) - T - 1) y_{\tau}^* \) (Note: The sum notation should be clear for the correct expression)

4. \( \rho_{t+1} = W_{t+1} / \text{tr}(W_{t+1}) \)

Output: \( \bar{y} = \frac{\delta \alpha}{R} e_1 + \frac{1}{T} \sum_{t=1}^{T} y_t^* \)  \quad e_1 = (1, 0, \ldots, 0)

\textbf{Thm (Arora-Kale '07)} \quad \bar{y}.b \leq (1+\delta) \alpha

Can find optimal value by binary search
Why Arora-Kale works?

Since \( y_t \in D_\alpha := \{y : y \geq 0, b.y \leq \alpha\} \)

\[
\overline{y}.b \leq \frac{\delta \alpha}{R} b_1 + \frac{1}{T} \sum_{t=1}^{T} y^t.b \leq (1 + \delta)\alpha
\]

Must check \( \overline{y} \) is feasible

From Oracle, for all \( t \):

\[
\text{tr} \left( \left( \sum_{j=1}^{m} y_{j}^t A_j - C \right) \rho^t \right) \geq 0
\]

We need:

\[
\lambda_{\min} \left( \left( \sum_{j=1}^{m} \left( \frac{1}{T} \sum_{t=1}^{T} y_{j}^t \right) A_j - C \right) \right) \geq 0
\]
Matrix Multiplicative Weight

**MMW** (Arora, Kale ‘07) Given $n \times n$ matrices $0 < M^t < I$ and $\varepsilon < \frac{1}{2}$,

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

with $\rho^t = \frac{\exp(-\varepsilon' \left( \sum_{\tau=1}^{t-1} M^\tau \right))}{\text{tr}(\ldots)}$ and $\varepsilon' = -\ln(1 - \varepsilon)$

$\lambda_n : \text{min eigenvalue}$

2-player zero-sum game interpretation:

- Player A chooses density matrix $X^t$
- Player B chooses matrix $0 < M^t < I$

Pay-off: $\text{tr}(X^t M^t)$

"$X^t = \rho^t$ strategy almost as good as global strategy"
**Matrix Multiplicative Weight**

**MMW (Arora, Kale ‘07)** Given $n \times n$ matrices $M^t$ and $\varepsilon < \frac{1}{2}$,

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

with

$$
\rho^t = \frac{\exp(-\varepsilon' \left( \sum_{\tau=1}^{t-1} M^\tau \right))}{\text{tr}(\ldots)}
$$

and

$$
\varepsilon' = - \ln(1 - \varepsilon)
$$

$\lambda_n$ : min eigenvalue

From Oracle:

$$
\text{tr} \left( \left( \sum_{j=1}^{m} y_j^t A_j - C \right) \rho^t \right) \geq 0
$$

By MMW:

$$
\lambda_{\min} \left( \left( \sum_{j=1}^{m} \left( \frac{1}{T} \sum_{t=1}^{T} y_j^t \right) A_j - C \right) \right) \geq 0
$$
Quantizing Arora-Kale Algorithm

We make it quantum as follows:

1. Implement ORACLE by Gibbs Sampling to produce $y^t$ and apply amplitude amplification to solve it in time $\tilde{O}(s^2 n^{1/2} m^{1/2})$

2. Sparsify $M^t$ to be a sum of $O(\log(m))$ terms:

   $$\overline{M}^t = \left( \|y^t\|_1 Q^{-1} \sum_{j=1}^{Q} A_{ij} - C + RI \right) / 2R$$

   $$(i_1, \ldots, i_Q) \sim y^t / \|y^t\|_1, \quad Q = O(\log(m))$$

3. Quantum Gibbs Sampling + amplitude amplification to prepare

   $$\overline{\rho}^t = \exp \left( -\varepsilon' \sum_{\tau=1}^{t} \overline{M}^T \right) / \text{tr}(\ldots)$$

   in time $\tilde{O}(s^2 n^{1/2})$.
Quantizing Arora-Kale Algorithm

We make it quantum as follows:

1. Implement ORACLE by Gibbs Sampling to produce $y^t$ and apply amplitude amplification to solve it in time $\tilde{O}(s^2 n^{1/2} m^{1/2})$

We’ll show there is a feasible $y^t$ of the form $y^t = Nq^t$ with

$q^t := \exp(h)/\text{tr}(\exp(h))$ and

$$h = \sum_{i=1}^{m} \left( \lambda \text{tr}(A_i \rho^t) + \mu b_i \right) |i\rangle \langle i|$$

We need to simulate an oracle to the entries of $h$. We do it by measuring $\rho^t$ with $A_i$.

To prepare each $\rho^t$ takes time $\tilde{O}(s^2 n^{1/2})$. To sample from $q^t$ requires $\tilde{O}(m^{1/2})$ calls to oracle for $h$. So total time is $\tilde{O}(s^2 n^{1/2} m^{1/2})$
Quantizing Arora-Kale Algorithm

We make it quantum as follows:

1. Implement ORACLE by Gibbs Sampling to produce $y^t$ and apply amplitude amplification to solve it in time $\tilde{O}(s^2 n^{1/2} m^{1/2})$

2. Sparsify $M^t$ to be a sum of $O(\log(m))$ terms:

$$M^t = \left( \|y^t\|_1 Q^{-1} \sum_{j=1}^Q A_{i_j} - C + RI \right) / 2R \quad \overline{M^t} \approx M^t$$

$$(i_1, \ldots, i_Q) \sim y^t / \|y^t\|_1, \quad Q = O(\log(m))$$

Can show it works by Matrix Hoeffding bound: $Z_1, \ldots, Z_k$ independent $n \times n$ Hermitian matrices s.t. $E(Z_i) = 0, \|Z_i\| < \lambda$. Then

$$\Pr \left( \left\| \frac{1}{k} \sum_{i=1}^k Z_i \right\| \geq \varepsilon \right) \leq n \cdot \exp \left( -\frac{k\varepsilon^2}{8\lambda^2} \right)$$
Quantum Arora-Kale, Roughly

Let $\rho^1 = I/n$, $\varepsilon = \frac{\delta \alpha}{2\omega R}$, $\varepsilon' = -\ln(1 - \varepsilon)$, $T = \frac{8\omega^2 R^2 \ln(n)}{\delta^2 \alpha^2}$.

For $t = 1, \ldots, T$

1. $y^t \leftarrow \text{ORACLE}(\rho^t)$

2. $M^t = \sum_{j=1}^{m} \left( y^t_j A_j - C + \omega I \right) / 2\omega$

3. Sparsify $M^t$ to $(M')^t$

4. $\rho^{t+1} = \exp \left( -\varepsilon' \left( \sum_{\tau=1}^{t} \left( (M')^\tau \right) \right) \right) / \text{tr}(\ldots)$

Gibbs Sampling

Output: $\overline{y} = \frac{\delta \alpha}{R} e_1 + \frac{1}{T} \sum_{t=1}^{T} y^t$
Implementing Oracle by Gibbs Sampling

\textbf{ORACLE}(\rho)

Searches for a vector \( y \) s.t.

i) \( y \in D_\alpha := \{y : y \geq 0, \ b.y \leq \alpha\} \)

\[ \sum_{j=1}^{m} \text{tr}(A_j \rho) y_j - \text{tr}(C \rho) \geq 0 \]
Implementing Oracle by Gibbs Sampling

Searches for (non-normalized) probability distribution $y$ satisfying two linear constraints:

$$\text{tr}(BY) \leq \alpha, \quad \text{tr}(AY) \geq \text{tr}(C\rho)$$

$$Y = \sum_i y_i |i\rangle\langle i|, \quad B = \sum_i b_i |i\rangle\langle i|, \quad A = \sum_i \text{tr}(A_i \rho) |i\rangle\langle i|$$

**Claim:** We can take $Y$ to be Gibbs: There are constants $N, \lambda, \mu$ s.t.

$$Y = N \frac{\exp(\lambda A + \mu B)}{\text{tr}(\ldots)}$$
Jaynes’ Principle

(Jaynes 57) Let $\rho$ be a quantum state s.t. $\text{tr}(\rho M_i) = c_i$

Then there is a Gibbs state of the form $\exp \left( \sum_i \lambda_i M_i \right) / \text{tr}(\ldots)$

with same expectation values.

**Drawback:** no control over size of the $\lambda_i$’s.
Finitary Jaynes’ Principle

(Lee, Raghavendra, Steurer ‘15) Let \( \rho \) s.t.

\[
\text{tr}(\rho M_i) = c_i
\]

Then there is a

\[
\sigma := \frac{\exp \left( \sum_i \lambda_i M_i \right)}{\text{tr}(\ldots)}
\]

with

\[
|\lambda_i| \leq 2 \ln(\text{dim}(\rho))/\varepsilon
\]

s.t.

\[
|\text{tr}(M_i \sigma) - c_i| \leq \varepsilon
\]

(Note: Used to prove limitations of SDPs for approximating constraints satisfaction problems; see James Lee’s talk)
Implementing Oracle by Gibbs Sampling

**Claim** There is a $Y$ of the form

$$Y = N \frac{\exp(\lambda A + \mu B)}{\text{tr}(\ldots)}$$

with $\lambda, \mu < \log(n)/\varepsilon$ and $N < \alpha$ s.t.

$$\text{tr}(BY) \leq \alpha + N\varepsilon, \quad \text{tr}(AY) \geq \text{tr}(C\rho) - N\varepsilon$$

$$Y = \sum_i y_i |i\rangle\langle i|, \quad B = \sum_i b_i |i\rangle\langle i|, \quad A = \sum_i \text{tr}(A_i \rho) |i\rangle\langle i|$$
Implementing Oracle by Gibbs Sampling

Claim There is a $Y$ of the form

$$ Y = N \frac{\exp(\lambda A + \mu B)}{\text{tr}(\ldots)} $$

with $\lambda, \mu < \log(n)/\varepsilon$ and $N < \alpha$ s.t.

$$ \text{tr}(BY) \leq \alpha + N\varepsilon, \quad \text{tr}(AY) \geq \text{tr}(C\rho) - N\varepsilon $$

Can implement oracle by exhaustive searching over $x, y, N$ for a Gibbs distribution satisfying constraints above

(only $\alpha \log^2(n)/\varepsilon^3$ different triples needed to be checked)
Conclusion and Open Problems

Quantum computers provide speed-up for SDPs

Many open questions:
- Can we improve the parameters (in terms of $R, r, \delta$)?
- Can we find optimal algorithm in terms of $n, m$ and $s$?
- Can we find relevant settings with superpoly speed-ups?
- Robustness to error?
- Q. computer only used for Gibbs Sampling. Application of small-sized q. computer?
Conclusion and Open Problems

Quantum computers provide speed-up for SDPs

Many open questions:
- Can we improve the parameters (in terms of $R$, $r$, $\delta$)?
- Can we find optimal algorithm in terms of $n$, $m$ and $s$?
- Can we find relevant settings with superpoly speed-ups?
- Robustness to error?
- Q. computer only used for Gibbs Sampling. Application of small-sized q. computer?

Thanks!