Quantum Speed-ups for Semidefinite Programming

Fernando G.S.L. Brandão

MSR -> Caltech

based on joint work with

Krysta Svore

MSR

Faculty Summit 2016

Quantum Algorithms

Exponential speed-ups:

Simulate quantum physics, factor big numbers (Shor's algorithm), ...,

Polynomial Speed-ups:

Searching (Grover's algorithm: $N^{1/2}$ vs $O(N)$), ...

Heuristics:

Quantum annealing (adiabatic algorithm), machine learning, ...

Quantum Algorithms

Exponential speed-ups:

Simulate quantum physics, factor big numbers (Shor's algorithm), ...,

Polynomial Speed-ups:

Searching (Grover's algorithm: $N^{1/2}$ vs $O(N)$), ...

Heuristics:

Quantum annealing (adiabatic algorithm), machine learning, ...

This Talk:

Solving Semidefinite Programming belongs here

Semidefinite Programming

… is an important class of convex optimization problems

$$\max \operatorname{tr}(CX)$$
$$\forall j \in [m], \qquad \operatorname{tr}(A_j X) \leq b_j$$
$$X \geq 0.$$

Input: n x n, r-sparse matrices C, A_1, ..., A_m and numbers b_1, ..., b_m
Output: X

Semidefinite Programming

… is an important class of convex optimization problems

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

Input: $n \times n$, r-sparse matrices $C, A_1, \ldots, A_m$ and numbers $b_1, \ldots, b_m$
Output: X

Some Applications: operations research (location probems, scheduling, ...), bioengineering (flux balance analysis, ...), approximating NP-hard problems (max-cut, ...), ...

Semidefinite Programming

... is an important class of convex optimization problems

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

Input: $n \times n$, r-sparse matrices $C, A_1, \ldots, A_m$ and numbers $b_1, \ldots, b_m$
Output: X

Some Applications: operations research (location probems, scheduling, ...), bioengineering (flux balance analysis, ...), approximating NP-hard problems (max-cut, ...), ...

Algorithms Interior points: $O((m^2nr + mn^2)\log(1/\varepsilon))$

Multiplicative Weights: $O((mnr\,(\omega R)/\varepsilon^2))$

"width" "size of solution"

Semidefinite Programming

... is an important class of convex optimization problems

$$
\begin{aligned}
& \max \operatorname{tr}(CX) \\
\forall j \in [m], \qquad & \operatorname{tr}(A_j X) \le b_j \\
& X \ge 0.
\end{aligned}
$$

Input: $n \times n$, r-sparse matrices $C, A_1, \ldots, A_m$ and numbers $b_1, \ldots, b_m$
Output: X

Some Applications: operations research (location probems, scheduling, ...), bioengineering (flux balance analysis, ...), approximating NP-hard problems (max-cut, ...), ...

Algorithms Interior points: $O((m^2nr + mn^2)\log(1/\varepsilon))$
Multiplicative Weights: $O((mnr\,(\omega R)/\varepsilon^2))$

Lower bound: No faster than $\Omega(nm)$, for constant ε, r, ω, R

Semidefinite Programming

$$\begin{aligned} &\max \operatorname{tr}(CX) \\ \forall j \in [m], \quad &\operatorname{tr}(A_j X) \le b_j \\ &X \ge 0. \end{aligned}$$

Normalization: $\|C\|, \|A_j\| \le 1$

We assume: $A_1 = I, b_1 = R, b_i = 1, i \neq 1$

Reduction optimization to decision:

$$\begin{aligned} &\max \operatorname{tr}(CX) \\ \forall j \in [m], \quad &\operatorname{tr}(A_j X) \le b_j \\ &X \ge 0. \end{aligned}$$

$\ge \alpha$

or

$\le \alpha + \delta$

SDP Duality

Primal: $\forall j \in [m],$

$$\max \operatorname{tr}(CX)$$
$$\operatorname{tr}(A_j X) \le b_j$$
$$X \ge 0.$$

Dual:

$$\min b.y$$
$$\sum_{j=1}^{m} y_j A_j \ge C$$
$$y \ge 0.$$

Quantum Algorithm for SDP

thm There is a quantum algorithm for solving SDPs running in time $\tilde{O}(n^{1/2} m^{1/2} r \delta^{-2} R^2)$

Quantum Algorithm for SDP

thm There is a quantum algorithm for solving SDPs running in time $\tilde{O}(n^{1/2} m^{1/2} r \delta^{-2} R^2)$

Input: n x n matrices r-sparse $C, A_1, \ldots, A_m$ and numbers $b_1, \ldots, b_m$

Output: Samples from $y/||y||_2$ and $||y||_2$

Q. Samples from $X/tr(X)$ and $tr(X)$

Primal:

$$\max \mathrm{tr}(CX)$$

$$\forall j \in [m], \quad \mathrm{tr}(A_j X) \leq b_j$$

$$X \geq 0.$$

Dual:

$$\min b.y$$

$$\sum_{j=1}^{m} y_j A_j \geq C$$

$$y \geq 0.$$

$\geq \alpha$

or

$\leq \alpha + \delta$

Quantum Algorithm for SDP

thm There is a quantum algorithm for solving SDPs running in time $\tilde{O}(n^{1/2} m^{1/2} r \delta^{-2} R^2)$

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

Output: Samples from $y/\|y\|_2$ and $\|y\|_2$

Q. Samples from $X/\mathrm{tr}(X)$ and $\mathrm{tr}(X)$

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:

$$|j, k, l, z\rangle \rightarrow |j, k, l, z \oplus (A_j)_{k f_{jk}(l)}\rangle \qquad f_{jk} : [r] \rightarrow [n]$$

choice of A_j row k l non-zero element

Quantum Algorithm for SDP

thm There is a quantum algorithm for solving SDPs running in time $\tilde{O}(n^{1/2} m^{1/2} r \delta^{-2} R^2)$

Classical lower bound At least time $\Omega(\max(n, m))$ for $r, \delta, R = \Omega(1)$. Reduction from Search

Quantum lower bound At least time $\Omega(\max(n^{1/2}, m^{1/2}))$ for $r, \delta, R = \Omega(1)$. Reduction from Search

Ex.

$\Omega(\max(m))$: $b_i = 1$, $C = I$

i) $A_j = I$ for all j (obj = 1)

ii) $A_j = 2I$ for random j in $[m]$ and $A_k = I$ for $k \neq j$ (obj = 1/2)

Quantum Algorithm for SDP

thm There is a quantum algorithm for solving SDPs running in time $\tilde{O}(n^{1/2} m^{1/2} r \delta^{-2} R^2)$

thm 2 There is a quantum algorithm for solving SDPs running in time $\tilde{O}(T_{Gibbs} m^{1/2} r \delta^{-2} R^2)$

$$T_{Gibbs} := \max_{\|\nu\|_\infty \leq 10 \log(n)/\delta^2} \mathrm{Time}\left(\exp(\nu_0 C + \sum_i \nu_i A_i)/Z\right)$$

If Gibbs states can be prepared quickly (e.g. by quantum metropolis), larger speed-ups possible

Quantum Algorithm for SDP

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

The Oracle

ORACLE(ρ)

Searches for a vector *y* s.t.

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

ii) $\sum_{j=1}^{m} \mathrm{tr}(A_j \rho) y_j - \mathrm{tr}(C\rho) \geq 0$

Dual: $\min b.y$

$$\sum_{j=1}^{m} y_j A_j \geq C$$

$$y \geq 0.$$

Width of SDP

$$\omega := \max_{y \in D_\alpha} \left\| \sum_j y_j A_j - C \right\| \leq \alpha \max_j \|A_j\| + \|C\|$$

$$\leq \alpha + 1$$

Arora-Kale Algorithm

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

For $t = 1, \ldots, T$

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

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

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

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

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

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

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 x n matrices M^t and $\varepsilon < 1/2$,

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

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

λ_n : 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: $\mathrm{tr}(X^t M^t)$

"$X^t = \rho^t$ strategy almost as good as global strategy"

Arora-Kale Algorithm

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

For $t = 1, \ldots, T$

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

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

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

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

Output: $\bar{y} = \frac{\delta\alpha}{R} e_1 + \frac{1}{T}\sum_{t=1}^{T} y^t$

Arora-Kale Algorithm

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

For $t = 1, \ldots, T$

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

2.

How to implement the Oracle?

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

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

Output: $\bar{y} = \frac{\delta\alpha}{R} e_1 + \frac{1}{T}\sum_{t=1}^{T} y^t$

Implementing Oracle by Gibbs Sampling

$\text{ORACLE}(\rho)$ Searches for a vector *y* s.t.

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

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

Implementing Oracle by Gibbs Sampling

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

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

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

claim: We can take Y to be Gibbs: There are constants *N, x, y* s.t.

$$Y = N \exp\left(xA + yB\right)$$

Jaynes' Principle

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

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

with same expectation values.

Drawback: no control over size of the λ_i's.

Finitary Jaynes' Principle

(Lee, Raghavendra, Steurer '15) Let ρ s.t. $\mathrm{tr}(\rho M_i) = c_i$

Then there is a $\sigma := \dfrac{\exp\left(\sum_i \lambda_i M_i\right)}{\mathrm{tr}(\ldots)}$

with $|\lambda_i| \leq 2\ln(n)/\varepsilon$

s.t. $|\mathrm{tr}(M_i\sigma) - c_i| \leq \varepsilon$

(Note: Used to prove limitations of SDPs for approximating constraints satisfaction problems)

Implementing Oracle by Gibbs Sampling

Claim There is a Y of the form $Y = N\frac{\exp(xA + yB)}{\mathrm{tr}(\ldots)}$

with x, y < log(n)/ε and N < α s.t.

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

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

N < α: $y \in D_\alpha := \{y : y \geq 0, \; b.y \leq \alpha\}$

$$\sum_i y_i \leq R y_1 + \sum_{i>1} y_i \leq \alpha$$

Implementing Oracle by Gibbs Sampling

Claim There is a Y of the form $Y = N \frac{\exp(xA + yB)}{\mathrm{tr}(\ldots)}$

with x, y < log(n)/ε and N < α s.t.

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

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)

Arora-Kale Algorithm

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. $y^t \leftarrow \mathsf{ORACLE}(\rho^t)$ Using Gibbs Sampling

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

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

4. $\rho^{t+1} = W^{t+1}/\mathrm{tr}(W^{t+1})$ Again, it's Gibbs Sampling

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

Quantum Speed-ups for Gibbs Sampling

(Poulin, Wocjan '09, Chowdhury, Somma '16) Given a r-sparse D x D Hamiltonian H, one can prepare exp(H)/tr(…) to accuracy ε on a quantum computer with $O(D^{1/2}\, r\, ||H||\, \mathrm{polylog}(1/\varepsilon))$ calls to an oracle for the entries of H

Based on amplitude amplification

$$\sum_i |\psi_i\rangle|\psi_i\rangle \quad \rightarrow \quad \sum_i |\psi_i\rangle|\psi_i\rangle|E_i\rangle$$

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

Quantizing Arora-Kale

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. $y^t \leftarrow \mathsf{ORACLE}(\rho^t)$ Use Gibbs Sampling

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

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

4. $\rho^{t+1} = W^{t+1}/\mathrm{tr}(W^{t+1})$ Again, Gibbs Sampling

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

Quantizing Arora-Kale

Problem: M^t is not sparse.

Solution: Sparsify it using operator Chernoff bound

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

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

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

Again, Gibbs Sampling

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

Sparsification

Suppose $Z_1, \ldots, Z_k$ are independent n x n Hermitian matrices s.t. $E(Z_i)=0$, $||Z_i||<\lambda$. Then

$$\Pr\left(\left\|\frac{1}{k}\sum_{i=1}^{k} Z_i\right\| \geq \varepsilon\right) \leq n.\exp\left(-\frac{k\varepsilon^2}{8\lambda^2}\right)$$

Applying to $M^t = \frac{\|y^t\|_1}{2\omega}\sum_{j=1}^{m} \overline{y}_j^t A_j - C/2\omega + I/2$

Sampling $j_1, \ldots, j_k$ from $\overline{y}^t = y^t/\|y^t\|_1$ with $k = O(\log(n)/\varepsilon^2)$

$$\left\|\frac{1}{k}\sum_{i=1}^{k} A_{j_i} - \sum_{j=1}^{m} \overline{y}_j^t A_j\right\| \leq \varepsilon$$

Quantizing Arora-Kale

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. $y^t \leftarrow \mathsf{ORACLE}(\rho^t)$ Use Gibbs sampling
2. $M^t = \sum_{j=1}^{m} (y_j^t A_j - C + \omega I)/2\omega$ Sparsify by random sampling
3. $W^{t+1} = \exp\left(-\varepsilon' \left(\sum_{\tau=1}^{t} M^\tau\right)\right)$
4. $\rho^{t+1} = W^{t+1}/\mathrm{tr}(W^{t+1})$ Again, Gibbs sampling

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

Quantizing Arora-Kale

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 $t = 1, \ldots, T$

Takes Õ($n^{1/2}r$) time on a quantum computer

2. $M^t = \ldots A_j - C + \omega I)/2\omega$ Sparsify by random sampling

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

4. $\rho^{t+1} = W^{t+1}/\mathrm{tr}(W^{t+1})$ Again, Gibbs sampling

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

Quantizing Arora-Kale

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

Use Gibbs sampling

ng

Needs to prepare: $\exp(\sum_i h_i \, |i\rangle\langle i|)/\mathrm{tr}(\ldots)$ with $h_i = x\,\mathrm{tr}(A_i\,\rho^t) + y\,b_i$

Can do quantumly with $\tilde{O}(m^{1/2})$ calls to an oracle to *h*.

Each call to *h* consists in estimating the value of $\mathrm{tr}(A_i\,\rho^t)$.

To estimate needs to prepare copies of ρ^t, which costs $\tilde{O}(n^{1/2}r)$

Overall: $\tilde{O}(n^{1/2}m^{1/2}r)$

Conclusion and Open Problems

We showed quantum computers can speed-up the solution of SPDs

One application is to solve the Goesman-Williamson SDP for Max-Cut in time $\tilde{O}(|V|)$. Are there more applications?

- Can we improve the parameters?
- Can we find (interesting) instances where there are superpolynomial speed-ups?
- Quantum speed-up for interior-point methods?

Thanks!