A Semidefinite Programming Formulation of Quantum and Classical Oracle Problems

Mike Mullan and Emanuel Knill
University of Colorado at Boulder and the
National Institute of Standards and Technology

Outline

1. Semidefinite Programming Formulation of Quantum Computation *[2,3]*
   - Can calculate **exact** error probabilities for a quantum computer optimally solving arbitrary quantum oracle problems
2. Remote State Preparation, and a Generalized Objective
3. Restricted Computation
   - Decoherence and analyzing a computation subject to noise
   - Optimal algorithms using reduced quantum resources (briefly)
4. Quantum Clocks

Quantum Algorithms, Generically

Goal: Minimize the number of queries of a quantum query algorithm via a generalization of Ambainis' adversary method *[1] (and recent progress given in [8])*

Try to express a quantum query algorithm as a semidefinite program (SDP):

Semidefinite Programming: Linear programming with semidefinite constraints

Quantum Query Algorithm: Arbitrary unitary operators interspersed with oracle operators acting on some initial state, followed by a measurement

$$|\psi(t)\rangle = U_t \Omega U_{t-1} \Omega \ldots \Omega U_0 |\psi(0)\rangle$$

Grover's Algorithm

Goal: Search – Find the marked element

1. $|\psi(0)\rangle = |0\rangle$
2. $U_0|\psi(0)\rangle = \frac{1}{2}\left(|0\rangle + |1\rangle + |2\rangle + |3\rangle\right)$
3. $\Omega U_0|\psi(0)\rangle = \frac{1}{2}\left((-1)^{\Omega_0}|0\rangle + (-1)^{\Omega_1}|1\rangle + (-1)^{\Omega_2}|2\rangle + (-1)^{\Omega_3}|3\rangle\right)$
4. $\Omega U_0|\psi(0)\rangle = \frac{1}{2}\left(|0\rangle - |1\rangle + |2\rangle + |3\rangle\right)$
5. $U_1\Omega U_0|\psi(0)\rangle = |1\rangle$

U = Inversion about the Mean $a_i \rightarrow -a_i + 2 * \langle a \rangle$

$$\Omega|\psi(t)\rangle = \sum_i a_i \Omega|i\rangle \rightarrow \sum_i a_i (-1)^{\Omega_i}|i\rangle$$

Ω_0	Ω_1	Ω_2	Ω_3
0	1	0	0

Other Oracles Are Possible

We can also have:

$\Omega(0)$	1	0	0	0
$\Omega(1)$	0	1	0	0
$\Omega(2)$	0	0	1	0
$\Omega(3)$	0	0	0	1

Assume each oracle can be given to our computer with some probability

New Goal: Find the marked element $\rightarrow$ Identify the applied oracle

The Computer

$$\rho^Q = \sum_{i,j} a_{i,j} |i\rangle\langle j|$$

Q: Querier system

A: Ancilla

O: Oracle system

$$\rho^O = \sum_{x,y} \sqrt{p_x}\sqrt{p_y}|x\rangle\langle y|$$

Pure state over possible oracles

The Computer

$$\rho^Q = \sum_{i,j} a_{i,j} |i\rangle\langle j|$$

Q: Querier system

A: Ancilla

$$\Omega^{OQ} = \sum_x |x\rangle\langle x| \Omega^Q(x)$$

O: Oracle system

$$\rho^O = \sum_{x,y} \sqrt{p_x}\sqrt{p_y} |x\rangle\langle y|$$

Pure state over possible oracles

The Computer

$$\rho^Q = \sum_{i,j} a_{i,j} |i\rangle\langle j|$$

Q: Querier system

A: Ancilla

$$\Omega^{OQ} = \sum_x |x\rangle\langle x| \Omega^Q(x)$$

$$\Omega^{OQ}(|x\rangle^O \otimes |\psi\rangle^Q) \rightarrow (|x\rangle^O \otimes \Omega^Q(x)|\psi\rangle^Q)$$

O: Oracle system

$$\rho^O = \sum_{x,y} \sqrt{p_x}\sqrt{p_y}|x\rangle\langle y|$$

Pure state over possible oracles

The Initial Joint Density Matrix

$$\rho^{OQ}(0) = \sum_{x,y} \sum_{i,j} \sqrt{p_x}\sqrt{p_y}|x\rangle\langle y| \otimes a_{i,j}|i\rangle\langle j|$$

$$\rho^{OQ}(0) = \begin{pmatrix} \sqrt{p_0}\sqrt{p_0}\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix} & \sqrt{p_0}\sqrt{p_1}\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix} \\ \sqrt{p_1}\sqrt{p_0}\boxed{\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix}} & \ddots \end{pmatrix}$$

$$\boxed{\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix}} \longrightarrow \rho^{Q}(0)$$

The Initial Joint Density Matrix

$$\rho^{OQ}(0) = \sum_{x,y}\sum_{i,j}\sqrt{p_x}\sqrt{p_y}|x\rangle\langle y| \otimes a_{i,j}|i\rangle\langle j|$$

$$\rho^{OQ}(0) = \begin{pmatrix} \sqrt{p_0}\sqrt{p_0}\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix} & \sqrt{p_0}\sqrt{p_1}\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix} \\ \sqrt{p_1}\sqrt{p_0}\boxed{\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix}} & \ddots \end{pmatrix}$$

$$\boxed{\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix}} \rightarrow \rho^{Q}(0)$$

$$\Omega^\dagger\rho^{OQ}(0)\Omega \rightarrow \sum_{x,y}\sum_{i,j}\sqrt{p_x}\sqrt{p_y}|x\rangle\langle y| \otimes \Omega(x)^\dagger a_{i,j}|i\rangle\langle j|\Omega(y)$$

$$\rho^{OQ} = \begin{pmatrix} \sqrt{p_0}\sqrt{p_0}\;\Omega(0)^\dagger\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix}\Omega(0) & \sqrt{p_0}\sqrt{p_1}\;\Omega(0)^\dagger\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix}\Omega(1) \\ \sqrt{p_1}\sqrt{p_0}\;\Omega(1)^\dagger\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & \ddots \end{pmatrix}\Omega(0)^\dagger & \ddots \end{pmatrix}$$

Quantum Algorithm → SDP (1)

Need both an objective and a set of linear constraints over ρ^{OQ}

(Straightforward) **Constraints**

$$\rho^{OQ}(t) \geq 0, \quad \rho^{O}(t) \geq 0$$

$$\rho^{O}(t) = tr_Q(\rho^{OQ}(t))$$

$$\rho^{O}(0) = \rho_0$$

$$t \in \{0 \dots t_f\}$$

Quantum Algorithm → SDP (2)

$$|\psi(t)\rangle = U_t \Omega U_{t-1} \Omega ... \Omega U_0 |\psi(0)\rangle$$

$$\rho^O(t) = tr_{QA}(U_t^{\dagger QA} \Omega^{\dagger OQ} \rho^{OQA}(t-1) \Omega^{OQ} U_t^{QA})$$

Quantum Algorithm → SDP (2)

$$|\psi(t)\rangle = U_t \Omega U_{t-1} \Omega ... \Omega U_0 |\psi(0)\rangle$$

$$\rho^O(t) = tr_{QA}(U_t^{\dagger QA} \Omega^{\dagger OQ} \rho^{OQA}(t-1) \Omega^{OQ} U_t^{QA})$$

$$\rho^O(t) = tr_{QA}(U_t^{QA} U_t^{\dagger QA} \Omega^{\dagger OQ} \rho^{OQA}(t-1) \Omega^{OQ})$$

Quantum Algorithm → SDP (2)

$$|\psi(t)\rangle = U_t \Omega U_{t-1} \Omega \ldots \Omega U_0 |\psi(0)\rangle$$

$$\rho^O(t) = tr_{QA}(U_t^{\dagger QA} \Omega^{\dagger OQ} \rho^{OQA}(t-1) \Omega^{OQ} U_t^{QA})$$

$$\rho^O(t) = tr_{QA}(U_t^{QA} U_t^{\dagger QA} \Omega^{\dagger OQ} \rho^{OQA}(t-1) \Omega^{OQ})$$

$$\rho^O(t) = tr_Q(\Omega^{\dagger OQ} \rho^{OQ}(t-1) \Omega^{OQ})$$

Quantum Algorithm → SDP (2)

$$|\psi(t)\rangle = U_t \Omega U_{t-1} \Omega \ldots \Omega U_0 |\psi(0)\rangle$$

$$\rho^O(t) = tr_{QA}(U_t^{\dagger QA} \Omega^{\dagger OQ} \rho^{OQA}(t-1) \Omega^{OQ} U_t^{QA})$$

$$\rho^O(t) = tr_{QA}(U_t^{QA} U_t^{\dagger QA} \Omega^{\dagger OQ} \rho^{OQA}(t-1) \Omega^{OQ})$$

$$\rho^O(t) = tr_Q(\Omega^{\dagger OQ} \rho^{OQ}(t-1) \Omega^{OQ})$$

Leads to the SDP S_E, corresponding to the application of oracle and unitary operators as above

$$S_E = \begin{cases} \rho^{OQ}(t) \geq 0, \quad \rho^O(t) \geq 0 \\ \rho^O(t) = tr_Q(\rho^{OQ}(t)) \\ \rho^O(t) = tr_Q(\Omega^{\dagger OQ} \rho^{OQ}(t-1) \Omega^{OQ}) \\ \rho^O(0) = \rho_0 \\ t \in \{0 \ldots t_f\} \end{cases}$$

Measurement and Remote State Preparation

The following protocol describes the measurement process at the end of the computation at time t_f. *[6]*

1. Q makes a measurement with any complete POVM $\{P_i^{QA}\}_i$

Measurement and Remote State Preparation

The following protocol describes the measurement process at the end of the computation at time t_f. *[6]*

1. Q makes a measurement with any complete POVM $\{P_i^{QA}\}_i$
2. Conditional on measurement outcome *i,* the oracle system state is (up to normalization)

$$\sigma_i = tr_{QA}(P_i^{QA}\rho^{OQA}(t_f))$$

Measurement and Remote State Preparation

The following protocol describes the measurement process at the end of the computation at time t_f. *[6]*

1. Q makes a measurement with any complete POVM $\{P_i^{QA}\}_i$
2. Conditional on measurement outcome *i,* the oracle system state is (up to normalization)

$$\sigma_i = tr_{QA}(P_i^{QA}\rho^{OQA}(t_f))$$

3. We measure a cost function A_i on this remotely prepared state, which indicates how successful our algorithm was

Measurement and Remote State Preparation

The following protocol describes the measurement process at the end of the computation at time t_f. *[6]*

1. Q makes a measurement with any complete POVM $\{P_i^{QA}\}_i$
2. Conditional on measurement outcome *i*, the oracle system state is (up to normalization)

   $$\sigma_i = tr_{QA}(P_i^{QA}\rho^{OQA}(t_f))$$

3. We measure a cost function A_i on this remotely prepared state, which indicates how successful our algorithm was

Q's POVM is unconstrained, but the set of states that can be prepared on O is restricted by: $\sum_i \sigma_i^O = \rho^O(t_f)$

The Complete SDP

$$S_M = \begin{cases} Minimize \ \sum_i tr(\sigma_i A_i) \ subject \ to: \\ \forall_i \sigma_i \geq 0 \\ \sum_i \sigma_i^O = \rho^O(t_f) \end{cases}$$

$$S_E = \begin{cases} \rho^{OQ}(t) \geq 0, \quad \rho^O(t) \geq 0 \\ \rho^O(t) = tr_Q(\rho^{OQ}(t)) \\ \rho^O(t) = tr_Q(\Omega^{\dagger OQ} \rho^{OQ}(t-1) \Omega^{OQ}) \\ \rho^O(0) = \rho_0 \\ t \in \{0 \ldots t_f\} \end{cases}$$

$$\mathrm{S} = \mathrm{S}_M \cup S_E$$

Conventional Cost Function

- Each measurement outcome *i* corresponds to the querier's guess of which oracle was applied
  - Search: *i = 2* – The marked element is in position 2

$$(\sigma_i)_{x,x} = P(measured\ i \cap oracle\ = \ x)$$

- Average Probability of Error: Probability that you had oracle *x*, measured outcome *i*, and that you shouldn't measure outcome *i* on oracle *x*
  - Search: *x = 1000, i = 2*

$$\epsilon = \sum_i \sum_{x:x\neq i} (\sigma_i)_{x,x} \qquad \langle x|A_i|x\rangle = 1 - \delta(i,x)$$

- This error probability will be **exact**. Q is free to choose the POVM which optimizes the cost function, and the SDP will find this optimum subject to its constraints.

Restricted Computation

- All experimentally available quantum computers are subject to noise
- Decoherence can reduce or eliminate any quantum advantage.
- All current, general quantum lower bound methods assume perfect quantum computation.
- We would like a way to characterize how “quantum” our device is

Decoherence

After each query, the environment will interact with the computer and will decohere the querier *[11]*

Post Unitary, Pre-Query

$$|\psi(0)\rangle^{OQE} = \sum_{x,i} a^{x}_{x,i} |x\rangle^{O} |i\rangle^{Q} |\epsilon_0\rangle^{E_1}$$

Decoherence

After each query, the environment will interact with the computer and will decohere the querier *[11]*

Post Unitary, Pre-Query

$$|\psi(0)\rangle^{OQE} = \sum_{x,i} a^x_{x,i} |x\rangle^O |i\rangle^Q |\epsilon_0\rangle^{E_1}$$

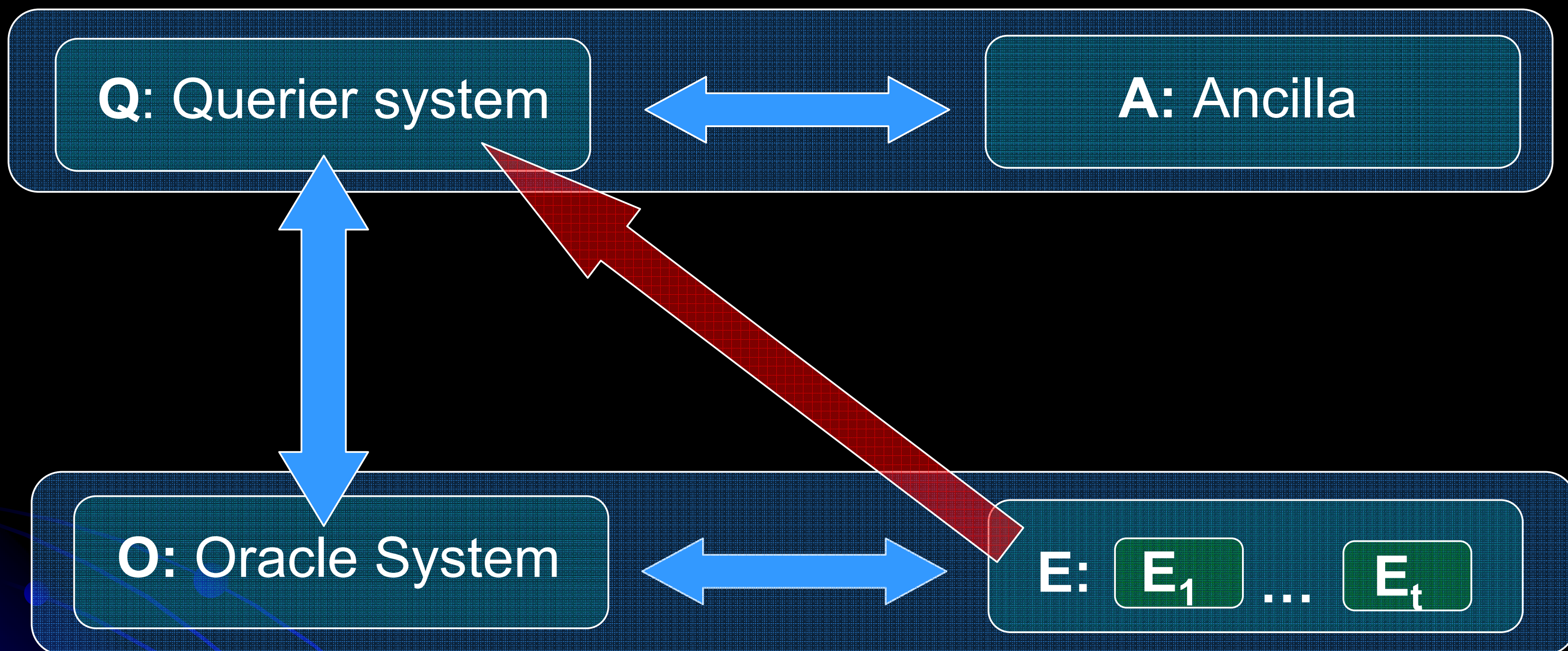
Post Query

$$|\psi(1)\rangle^{OQE} = \sum_{x,i} a^x_{i,a} |x\rangle^O (-1)^{\Omega(x)_i} |i\rangle^Q |\epsilon_i\rangle^{E_1} |\epsilon_0\rangle^{E_2}$$

The size of the environment grows with every query, so at time t:

$$E(t) \equiv E = E_1 \otimes E_2 \ldots \otimes E_t$$

Adding the Environment



The environment is part of the oracle system that the querier has no access to, but acts on the querier during each query.

The Extended SDP

Minimize $tr(\sum_i \sigma_i A_i)$ subject to:

$$\forall_i \sigma_i^{OE} \geq 0$$

$$\rho^{OQE}(t) \geq 0, \quad \rho^{OE}(t) \geq 0$$

$$\sum_i \sigma_i^{OE} = \rho^{OE}(T)$$

$$\rho^{OE}(t) = tr_Q(\rho^{OQE}(t))$$

$$\rho^{OE}(t) = tr_Q(D^{\dagger QE_t}\Omega^{\dagger OQ}\rho^{OQE}(t-1) \otimes |\epsilon_0\rangle_{E_t}^{E_t}\langle\epsilon_0|\Omega^{OQ}D^{QE_t})$$

$$\rho^{OE}(0) = \rho_0$$

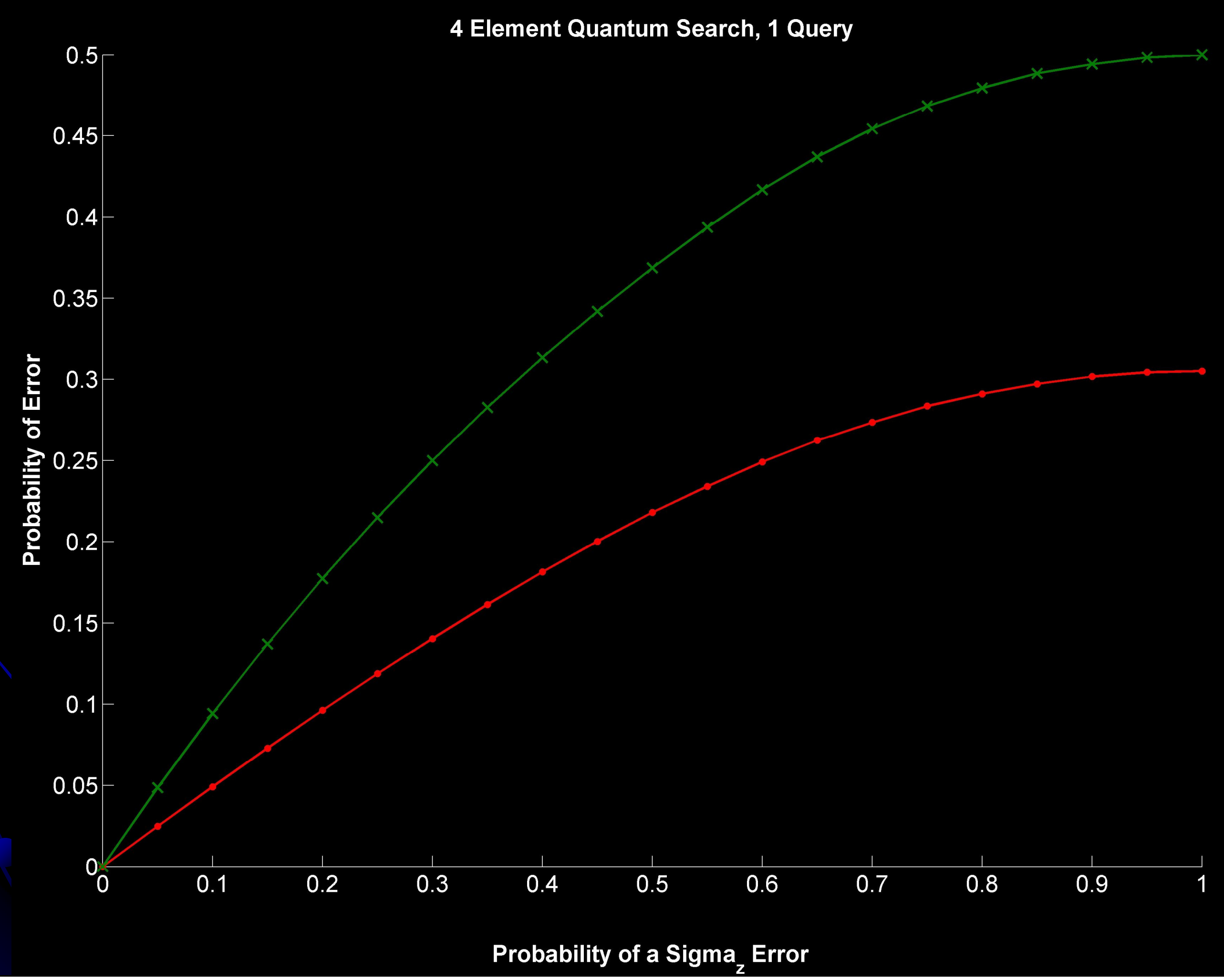
$$t \in \{0 \ldots t_f\}$$

Quantum/Classical Interpolation

One or more qubits may decohere independently with probability *p*

$$|x\rangle^O|i\rangle^Q|\epsilon_0\rangle^E \rightarrow \sqrt{1-p}|x\rangle^O(-1)^{\Omega(x)_i}|i\rangle^Q|\epsilon_0\rangle^E + \sqrt{p}|x\rangle^O(-1)^{\Omega(x)_i}|i\rangle^Q|\epsilon(i)\rangle^E$$

Red: One qubit decoherence
Green: Two qubit decoherence



Comments

- The size of the SDP grows rapidly, particularly when modeling decoherence. (< 10 qubits)
- Since the SDP outputs the optimal density matrix at every time step, we can reconstruct the inter-query unitaries, and hence the optimal algorithm
- We have recently explored these issues from an alternate angle by asking: *Is there an equally optimal algorithm that uses fewer quantum resources?*
  - Use an alternate SDP which tries to split the optimal algorithm into two or more parts that can be classically combined
  - Use the Von Neumann entropy to quantify the quantum resources needed in each branch
- See slides at end of talk for details

Clocks

- Our SDP formulation is highly general, and can describe quantum **processes** as well as quantum query algorithms
- Specifically, if an element drawn from a continuous set of unitary operators, indexed by some parameter, acts on a state, our formulation can be used to optimally estimate the value of this parameter
- We will use this idea to optimize the accuracy of quantum clocks
- **Goal**: Express the operation of the a quantum clock as a black box oracle problem.

Atomic Clock Basics

- Count the ticks of some periodic system and interpret as time.
- Atoms have a discrete set of energy levels and can transition between them by absorbing or emitting photons. ($E = hf$)
- Atomic clocks count the oscillations of a laser whose frequency is stabilized to some atomic transition
- 1 second ≡ 9,192,631,770 cycles of the radiation corresponding to a hyperfine transition in Cesium

NIST F1

Spectroscopy

- Consider a set of *N* ions and the set of states labeled by $|k\rangle$ which have *k* ions in the excited state and *N* - *k* ions in the ground state. (Dicke states)

- Over time, our laser/classical oscillator will drift from the atomic resonance. We need to measure and correct this detuning.

- After an interrogation period of length T with a laser tuned near resonance *[7]*

$$|k\rangle \rightarrow e^{-ik(f-f_0)T}|k\rangle$$

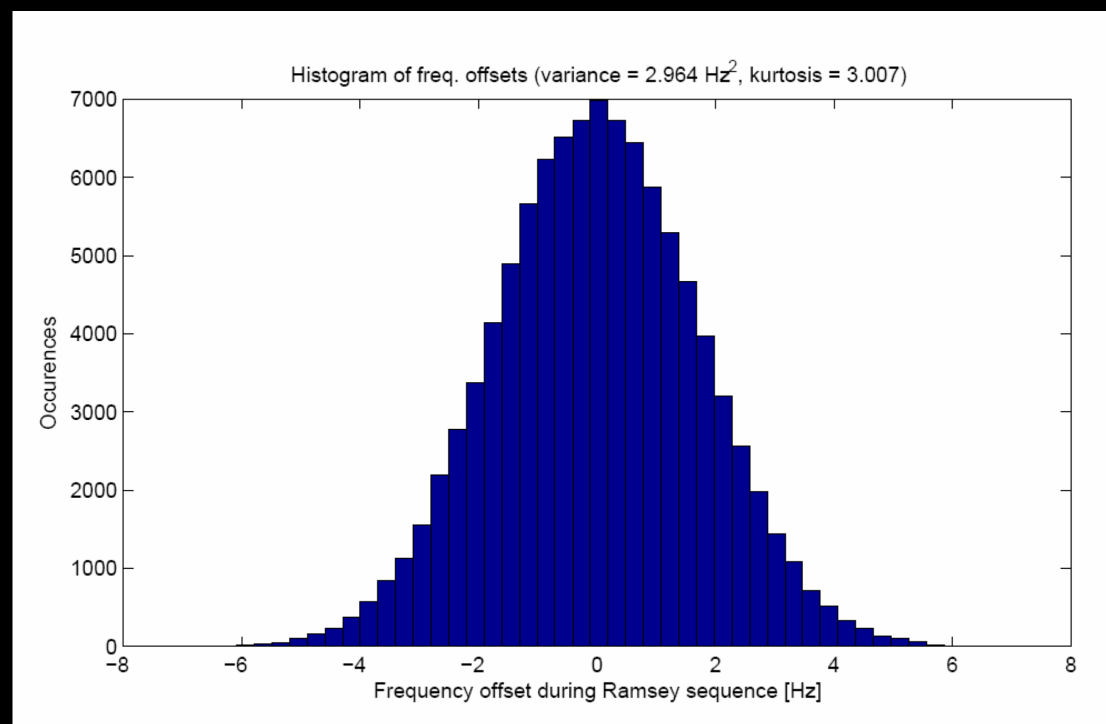
- This acts as our black box/oracle operation.

Goal: Phase Estimation

Frequency Priors and the Clock Oracle

Given $f - f_0$ we know that: $|k\rangle \to e^{-ik(f-f_0)T}|k\rangle$

We don't know $f - f_0$, but we can construct a prior probability distribution over $f - f_0$. Likewise, before, we didn't know which oracle our algorithm would be given, so we assigned each a probability.



Clock SDP

$$\rho^{Q} = \sum_{k,l} a_k |k\rangle\langle l|$$

$$\rho^{O} = \sum_{x,y} \sqrt{p_x}\sqrt{p_y}|x\rangle\langle y|$$

$$\Omega = \sum_{x} |x\rangle\langle x| e^{-i\Delta f_x \sigma_z T/2}$$

Goals:
1. Determine optimal initial state of Q
2. Determine optimal measurement

After one query

$$\Omega^{\dagger}\rho^{OQ}\Omega \rightarrow \sum_{x,y}\sum_{k,l} a_{x,y,k,l}|x\rangle\langle y| \otimes e^{i\Delta f_x kT}|k\rangle\langle l| e^{-i\Delta f_y lT}$$

Clock Cost Function

- Measure with a discrete, complete POVM $\{P_i^Q\}_i$. whose elements each correspond to a frequency guess, Δf_i
- Remote State Preparation:

$$\sigma_i = tr_Q(P_i^Q \rho^{OQ})$$

$$(\sigma_i)_{x,x} = P(measured\ \Delta f_i \cap frequency\ =\ \Delta f_x)$$

- Penalize guesses far from the true frequency

$$\langle \Delta f_x | A_i | \Delta f_x \rangle = (\Delta f_i - \Delta f_x)^2$$

- From the final density matrix and the remotely prepared mixed states, we can reconstruct this optimal POVM

Discussion

- Prior work has assumed a uniform probability distribution over clock oracles. *[4]*
- This technique is subject to discretization error
- Working on bounds – error appears small
- Al^+ Clock at NIST is a natural application of this technique. Uses only 1-2 ions and is the most accurate clock in the world (8.6×10^{-18}) *[5,9]*

Results

- Optimal initial states have equal amplitudes for *k* ions in the excited state and *N* – *k* in the ground state. e.g. for *N* = *3*:

$$|\psi_0\rangle = a_0\left(|0\rangle + |3\rangle\right) + a_1\left(|1\rangle + |2\rangle\right)$$

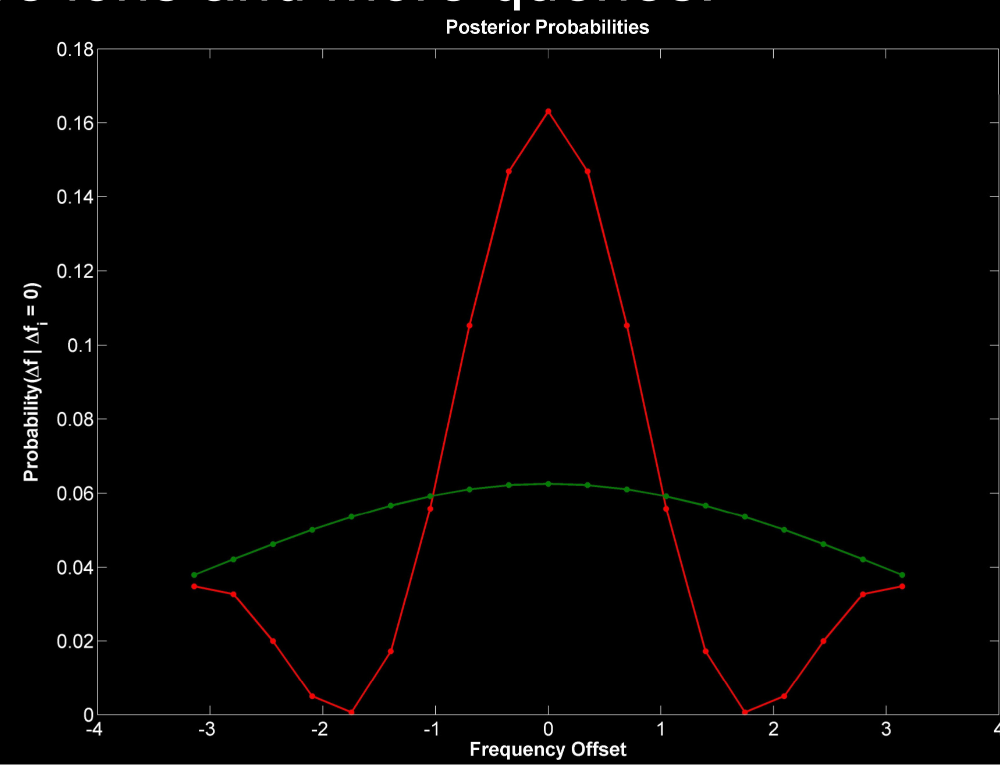
- One query on 2 ions is equivalent to 2 queries on one ion. Not necessarily true with more ions and more queries.

A narrower posterior corresponds to an increase in knowledge.

Green: Priors

Red: Posteriors

$P(\Delta f | \Delta f_i = 0)$



Conclusions

- This SDP formulation is a powerful and highly general way of describing quantum algorithms and quantum processes
- The size of the SDP grows quickly, but this technique remains applicable to many interesting problems.
- Here we analyzed quantum computers subject to decoherence, and developed a technique to examine the “quantumness” of a device.
- The application of our SDP to quantum clocks indicates that it likely has many other applications

References

1. A. Ambainis, *Quantum lower bounds by quantum arguments*
2. H. Barnum, M. Saks, and M. Szegedy, *Quantum query complexity and semidefinite programming*
3. H. Barnum, *Semidefinite programming characterization and spectral adversary method for quantum complexity with noncommuting unitary queries*
4. V. Buzek, R. Derka, and S. Massar, *Optimal quantum clocks*
5. C.W. Chou, et. al., *Frequency comparison of two high-accuracy Al^+ optical clocks*
6. L. Hughston, R. Jozsa, and W. Wooters, *A complete classification of quantum ensembles having a given density matrix*
7. N.F. Ramsey, *Molecular Beams*
8. B. Reichardt, Reflections for quantum query complexity: *The general adversary bound is tight for every boolean function*
9. P.O. Schmidt, et. al., *Spectroscopy using quantum logic*
10. B. Schumacher, *Quantum coding*
11. W. Zurek, *Decoherence, einselection and the quantum origins of the classical*

Algorithm Reconstruction

- SDP outputs a density matrix corresponding to the state of the computer at every timestep

- Purify the following pairs of states into A, where |A| = |O||Q|E|

$$D^{\dagger QE_t}\Omega^{\dagger QO}\rho^{OQE}(t)\Omega^{QO}D^{QE_t} \rightarrow \rho'^{OQEA}(t)$$

$$\rho^{OQE}(t+1) \rightarrow \rho^{OQEA}(t+1)$$

- Since we know that

$$\rho^{OQEA}(t+1) = U(t)^{\dagger QA}\rho'^{OQEA}(t)U(t)^{QA}$$

we can solve for the inter-query unitaries. Since the SDP yields the solution with the optimal probability of success, this reconstructs an optimal algorithm. (quantum or classical)

Measuring Quantum Resources

- Our SDP formulation will yield the algorithm with the optimal probability of success.
- But can we find an "alternate" algorithm with the same probability of success that uses fewer quantum resources?
- Quantify quantum resources via Von Neumann entropy

$$S(\rho) \equiv tr(\rho \; lg(\rho))$$

- By Schumacher's quantum coding, we know this is the minimum number of qubits needed to represent a quantum state *[10]*

Separate Computational Paths

- Let the optimal algorithm run normally until time t_s, then measure
- Quantify the quantum resources needed to reach timestep $t_s + 1$
  - Try to express $\rho^O(t_s)$ as the incoherent sum of $\rho_1^O(t_s),\ \rho_2^O(t_s) \ldots \rho_n^O(t_s)$
  - These independent computational paths can be classically combined.
  - Calculate entropy as:

$$S_T = tr(\rho_1^O) S\left(\frac{\rho_1^O}{tr(\rho_1^O)}\right) + tr(\rho_2^O) S\left(\frac{\rho_2^O}{tr(\rho_2^O)}\right) + \ldots$$

  - Since the full density matrix is pure: $S(\rho^O) = S(\rho^{QA})$

Splitting SDP

- Add the following constraints to a new SDP

$$\rho^{O}_{opt}(t) = \rho^{O}_{1}(t) + \rho^{O}_{2}(t)$$

$$tr_Q(\rho^{OQ}_{1}(t)) = \rho^{O}_{1}(t) \qquad t \geq t_s$$

$$tr_Q(\rho^{OQ}_{2}(t)) = \rho^{O}_{2}(t)$$

with the usual sets of constraints on these new density matrices. Here, the oracle density matrix is the optimal solution to our original SDP.

- Add an objective which maximizes the distance between the two subparts of the oracle density matrix.
- This will be a nonlinear objective so we use an iterative strategy

Splitting Results

- Recursively perform this splitting, calculating the entropy at each iteration.

8 Element Search/OR:

0. 2.056
2. 2.0096
4. 2.0093
8. 2.0093

4 Element PARITY

0. 2
2. 1
4. 0
8. 0

- Parity looks classical in the Hadamard basis.
- Entropy is a basis independent quantity.