# Quantum Computing (Fall 2013) Instructor: Shengyu Zhang.

# Lecture 6 Quantum query complexity: Upper bound.

In this lecture, we’ll first finish the proof for the quantum adversary method as a lower bound of quantum query complexity. Then we show a matching upper bound, by which we prove the result that the quantum adversary method is in the same order of the quantum query complexity.

## Lower bound: Quantum adversary method

Recall that the quantum query complexity is defined as the minimum number of queries needed for an -error quantum query algorithm to solve a computational problem on the worst case input. The best lower bound by the quantum adversary method is
where . We’ll first finish the proof of the following theorem.

## Upper bound:

It turns out that the above lower bound is tight. In this section, we’ll show an even strong result, a quantum query algorithm solving a more general computational task of *quantum state conversion* [LMR+11]. Suppose we have a set of pure quantum states and we’d like to convert them to another set of pure quantum states . (Usually the Greek letters and are for mixed states, but somehow they are used in [LMR+11] to denote pure states, and we just respect their notation here.) The conversion need to be correct for every , namely we need to use one universal algorithm to convert to for all simultaneously. In general this is not doable without the knowledge of , as shown by the following basic result.

**Exercise**. There is a unitary operation that converts the states to if and only if for all .

Since this condition doesn’t hold for general and . So we need to get some knowledge of the index . (In the extreme case, consider that we have the full knowledge of . Then we don’t even need the given state ---we can generate by ourselves.) We access by queries as usual. The question is what the minimum number of queries to is needed. We of course allow the algorithm to take ancilla, so the algorithm takes as input and generates a state close to for some . This generalizes the function evaluation setting, where all , and target states is for some ancilla .

The result is the following:

**Theorem**. The query complexity of the above state conversion problem is at most

where

* , ,
* with ,

For Boolean function evaluation, note that all and , thus and . It turns out that

by SDP duality. Thus the above theorem does implies the tightness of as a lower bound of the quantum query complexity.

Let . An optimum solution , in the definition of has the following properties by definition
 , , (1)
 ( for Boolean ) (2)

Instead of repeating the proof in the usual way, let me try to extract the line of main ideas.

* We actually only need to deal with the case , since otherwise we can first attach a to , and then transfer it to , and then discard the .
* To transfer to , it suffices to keep unchanged and flipping to , where
 , and .
(Indeed, .) So it’s enough to find a s.t. is close to the 1-eigenspace and is close to the -eigenspace.
* Actually doesn’t need to be close to -eigenspace. As long as doesn’t have a large support on eigenspace of corresponding to eigenvalue with small phases, a tool called *Phase Detection* can move to close to . (Namely, as long as moves most of ’s components, with respect to ’s eigenvectors, **away** in phase, not necessarily to the same far, the Phase Detection works.) More specifically, , , there is a circuit which, on **any** -eigenvector of where , has

 (3)

Here the number of extra qubits needed is , and number of queries of controlled- and controlled-is.

* The paper finds such a , which consists of repeated applications of two reflections.

In what follows, we only consider the Boolean-function evaluation case for simplicity.

**Algorithm**: Run Phase Detection on unitary and state , with precision and error .

* Suppose that , and is an optimal solution in the definition of .
* is the projection onto , where
 .
Note that here we attached another space to (where originally is), s.t. in the specification of the algorithm lives in the part of . So are states . But is in the whole space , with the first term in and second in . Note that is the direct sum, not product.
* requires that when the first register of is , the second register of be .

**Query complexity**:  . Each takes one query, and doesn’t need any query.

**Correctness**:
(Notation: is the projection onto the eigenspace of with eigenphase , and similarly for . That is, if is the spectral decomposition of , then
 and . )

* is close to a -eigenvector of . More precisely, .
**Fact 1.** .
[Proof] Actually is in the subspace and almost in the subspace . For the former, note that is in , while onlyhas requirement on . For the latter, directly bounding is not that obvious: , but there are many ’s with . Fortunately, one can add a little bit to to cancel out the annoying factor: Define , then by Eq.(2). And note that the extra term in is of -small norm because of Eq.(1). (Also note that is still in the subspace because the extra term exactly satisfies the requirement of .) □
* has a small support on small-eigenvectors of .
**Fact 2.** , .
[Proof] We’ll use a spectral gap lemma that is frequently used in one form or another.

**Lemma**. Suppose and are two projections, and . Suppose that is a complete orthonormal set of eigenvectors of , with the respective eigenvalues . For any vector , if , then for any , .

We’ll skip the proof of the lemma, and state how to use it to prove Fact 2. The application is very straightforward by setting . Note that and by definition of and , respectively. □

Now putting the above two together, we can show the correctness of the algorithm.

We use Fact 1 to bound the item 1, and use Eq.(3) and Fact 2 to bound item 2.

## Reference

[LMR+11] Troy Lee, Rajat Mittal, Ben Reichardt, Robert Spalek, Mario Szegedy**, Quantum query complexity of state conversion**, FOCS’11.