Relativized Separation of EQP from P^{NP}

Frederic Green* Randall Pruim†

Abstract

An oracle is constructed relative to which quantum polynomial time (EQP) is not polynomial-time Turing reducible to NP. That is, there is an A such that $\mathsf{EQP}^A \not\subseteq \mathsf{P}^{\mathsf{NP}^A}$. This generalizes and simplifies previous separations of EQP from NP and ZPP, due to Berthiaume and Brassard. A key element of the proof is the use of a special property of Grover's algorithm for database search, in order to show that the test language is in EQP^A.

Keywords: Computational Complexity, Theory of Computation, Quantum Computation.

1 Introduction

One of the primary goals of quantum computational complexity theory is to compare quantum complexity classes with classical complexity classes. A quantum class of central interest is EQP ("**E**xact **Q**uantum **P**olynomial" time [3]), the quantum analog of P. The classical complexity of EQP was already an object of study prior to Shor's seminal result that factoring can be done in probabalistic quantum polynomial time [7], and Grover's $O(\sqrt{N})$ quantum algorithm [6] for searching an unsorted database of N elements. However, to date there is no evidence that EQP and P are different (*proving* they are different would imply $\mathsf{P} \neq \mathsf{PSPACE}$), nor are there any interesting, natural problems known to be in EQP that are not known to be in P. Many researchers continue to wonder whether EQP is in the polynomial hierarchy. In light of the discovery [5] that NQP, the quantum analog of NP, is

*Department of Mathematics and Computer Science, Clark University, Worcester, MA 01610, fgreen@black.clarku.edu

†Department of Mathematics and Statistics, Calvin College, Grand Rapids, MI 49546, rpruim@calvin.edu

the same as $\mathsf{coC_=P}$, it is also reasonable to ask whether there is a natural characterization of EQP in terms of a classical complexity class.

Oracle separations can help this latter investigation, because they rule out certain relativizable relationships between EQP and other classes. For example, Berthiaume and Brassard [3] showed that there is an oracle relative to which $\mathsf{EQP} \not\subseteq \mathsf{NP} \cup \mathsf{coNP}$. This rules out the possibility that (for example) NP or ZPP contain EQP via a proof that relativizes.

In this note, we give a simple construction of an oracle relative to which EQP is not in $\mathsf{P^{NP}}$. This subsumes the result of [3] mentioned above. The novel element in the proof is not the diagonalization (which is similar to Stockmeyer's construction [8] of an oracle relative to which BPP is not in $\mathsf{P^{NP}}$), but rather showing that the diagonalizing set is in EQP. The key is to exploit a property of Grover's algorithm, pointed out by Boyer, Brassard, Høyer and Tapp [4]. In searching for a key that occurs in exactly 1/4 of the elements of a database, Grover's algorithm has success probability 1 after a single iteration.

2 Preliminaries

We assume familiarity with the basics of classical computational (e.g., [1]) complexity as well as quantum computation (e.g., [2]).

We consider a set to be a subset of Σ^* where $\Sigma = \{0,1\}$. If $A \subseteq \Sigma^*$, then $A(x)$ denotes the characteristic function of A (i.e., $A(x) = 1$ if $x \in A$ and otherwise $A(x) = 0$).

A *qubit* is a quantum mechanical state which is a linear superposition of the states $|0\rangle$ and $|1\rangle$. An n-bit *quantum register* is a superposition of tensor product states of the form $|x_1\rangle \otimes |x_2\rangle \cdots \otimes |x_n\rangle$ where each $x_i \in \Sigma$. If $|\Psi_1\rangle$ and $|\Psi_2\rangle$ are quantum states, we write $|\Psi_1\rangle|\Psi_2\rangle$ for the tensor product $|\Psi_1\rangle \otimes |\Psi_2\rangle$. For notational convenience, we also write $|x_1, ..., x_n\rangle$ or $|\mathbf{x}\rangle$ (so that $\mathbf{x}$ denotes the list of bits $x_1, ..., x_n$) in place of $|x_1\rangle \otimes |x_2\rangle \cdots \otimes |x_n\rangle$.

We take the quantum Turing machine as a model of quantum computation. The oracle access mechanism for quantum machines uses two quantum registers (one to hold the query string x and one to hold a qubit b where the query answer is recorded) and works as follows. Let A be a set, let $|x\rangle$ the contents of the quantum *query register*, and let $|b\rangle$ be the contents of the *query answer register*. When we want to query A, in one step, the state jumps from $|x\rangle|b\rangle$ to $|x\rangle|b \oplus A(x)\rangle$.

We recall some unitary transformations used by Grover in his algorithm [6]. The transformation W has the following effect on a single qubit:

$$W|b\rangle = \frac{1}{\sqrt{2}}\sum_{a=0}^{1}(-1)^{ab}|a\rangle.$$

It has the effect of a Walsh-Hadamard transform on an n-qubit register:

$$W|x_1, ..., x_n\rangle = \frac{1}{2^{n/2}}\sum_{\mathbf{y}\in\Sigma^n}(-1)^{\mathbf{x}\cdot\mathbf{y}}|y_1, ..., y_n\rangle,$$

where $\mathbf{x}\cdot\mathbf{y}$ is the inner product $\sum_{i=1}^{n} x_i y_i$.

When an oracle set A is present in a quantum computation, the tranformation Q on a state of the form $|x\rangle \otimes |b\rangle$ performs a query as described above. The most important use of this transformation below is with the query answer register in superposition, in states of the form $|x\rangle \otimes (|0\rangle - |1\rangle)$. This has the effect of reversing the sign of the state if the answer is "yes," which is similar to the transformation denoted F in Grover's paper. Finally, the transformation $\hat{F}$ on a quantum register of n bits flips the sign of any state except the one containing n 0's.

3 The Oracle

Let

$$L(A) = \{0^n | \text{the number of strings in } A \text{ of length } n \text{ is } \tfrac{1}{4}2^n\}\ .$$

Consider oracles A such that for any n, there are either $\frac{1}{4}2^n$ or $\frac{3}{4}2^n$ strings of length n in A. We call such an oracle *symmetric*. The proof of our main result follows from two lemmas. The first Lemma says that a symmetric A can be constructed such that $L(A) \notin \mathsf{P}^{\mathsf{NP}^A}$. The second shows that for any symmetric A, $L(A) \in \mathsf{EQP}^A$.

Lemma 3.1 *There is a symmetric oracle A such that $L(A) \notin \mathsf{P}^{\mathsf{NP}^A}$.*

Proof: Let $\{N_1, N_2, N_3, ...\}$ be an enumeration of NP oracle machines. Define the canonical NP^A-complete language,

$$K(A) = \{\langle x, k, 0^t\rangle | \mathsf{NP}^A\text{-machine } N_k^A \text{ accepts } x \text{ in } t \text{ time steps}\}\ .$$

Since any langauge in $\mathsf{P}^{\mathsf{NP}^A}$ is accepted by polynomial time machine using oracle $K(A)$, an enumeration $\{M_1, M_2, M_3, ...\}$ of all polynomial time oracle machines provides us with an implicit enumeration of all $\mathsf{P}^{\mathsf{NP}^A}$ languages.

The construction of A is in stages. At stage i, we pick an n_i and put a set of strings of length n_i in A such that $0^{n_i} \in L(A)$ iff M_i^A rejects 0^{n_i}.

Stage i works roughly as follows. Choose n_i to be larger than the length of any query made in previous stages Run M_i on input 0^{n_i} up to its first query to $K(A)$. Suppose the first query is $\langle y, k, 0^t \rangle$. If there is any way to add strings of length n_i to A in order that N_k^A accepts y in time t, then do so. Since N_k only requires one accepting path in order to accept, this fixes a polynomial number of elements in $A^{=n_i}$. If there is no way to make N_k accept, then it is fixed to reject no matter what we do with the oracle. The first query in M_i's computation is now "frozen." Do the same with all subsequent queries in M_i's computation: freeze the answer to the query if possible, keeping all previous queries frozen. This can be done for all the queries that M_i makes by fixing at most polynomially many strings of length n_i to be in A. Suppose q such elements have been fixed. After all queries are frozen, determine if M_i rejects. If it does, put $\frac{1}{4}2^{n_i} - q$ of the remaining strings of length n_i in A_i. If, on the other hand, M_i accepts, put $\frac{3}{4}2^{n_i} - q$ strings of length n_i in A_i.

It is clear that A is symmetric and that $L(A) \notin \mathsf{P}^{\mathsf{NP}}$. $\square$

Lemma 3.2 *For any symmetric A, $L(A) \in \mathsf{EQP}^A$.*

Proof:

The main idea is that the language $L(A)$ can be decided in EQP^A for any symmetric A by applying one iteration of Grover's algorithm.

We describe an EQP^A-machine M^A that recognizes $L(A)$. M has one register to hold the input x, one query register and one query answer register. In the beginning the state looks like $|\Psi_0\rangle = |x\rangle|\mathbf{0}\rangle|1\rangle$, where the $\mathbf{0}$ denotes n zeroes (where $n = |x|$). M works as follows. First, it checks to see if x is of the form 0^n. If not, it rejects (with probability 1, of course). Otherwise, we evolve the state according to one iteration of Grover's algorithm, that is, applying the transformation $W\hat{F}WQW$ to the state $|x\rangle|\mathbf{0}\rangle|1\rangle$.

We first take $|\Psi_0\rangle$ and apply W to the query and query answer registers (which start out as all zeroes and a one, respectively). We get,

$$W|\Psi_0\rangle = \frac{1}{2^{(n+1)/2}} \sum_{z \in \{0,1\}^n} |x\rangle|z\rangle(|0\rangle - |1\rangle).$$

Now apply the transformation Q (i.e., make a query to A), and apply W again. Writing out these two steps, we obtain,

$$\begin{aligned}|\Psi_0\rangle &\mapsto \frac{1}{2^{(n+1)/2}} \sum_{z\in\{0,1\}^n} (-1)^{A(z)}|x\rangle|z\rangle(|0\rangle - |1\rangle) \\ &\mapsto \frac{1}{2^{n+1/2}} \sum_{y\in\{0,1\}^n} \sum_{z\in\{0,1\}^n} (-1)^{A(z)}(-1)^{y\cdot z}|x\rangle|y\rangle(|0\rangle - |1\rangle)\end{aligned}$$

Now apply $\hat{F}$ to the query register. This flips the sign of all states except the one with $y = 0^n$. For ease in notation, suppress the input and the query answer bit for now (that is, the $|x\rangle$ and $(|0\rangle - |1\rangle)$, which at this point are unaffected by the computation anyway). Then applying $\hat{F}$ we obtain,

$$\begin{aligned}|\Psi_0\rangle &\mapsto \frac{1}{2^{n+1/2}} \sum_{z\in\{0,1\}^n} \left[(-1)^{A(z)}|\mathbf{0}\rangle - (-1)^{A(z)} \sum_{y\neq\mathbf{0}} (-1)^{y\cdot z}|y\rangle\right] \\ &= \frac{1}{2^{n+1/2}} \sum_{z\in\{0,1\}^n} \left[((-1)^{A(z)} + (-1)^{A(z)})|\mathbf{0}\rangle - (-1)^{A(z)} \sum_{y\in\{0,1\}^n} (-1)^{y\cdot z}|y\rangle\right] \\ &= \frac{2}{2^{n+1/2}} \sum_{z\in\{0,1\}^n} (-1)^{A(z)}|\mathbf{0}\rangle - \frac{1}{2^{n+1/2}} \sum_{z\in\{0,1\}^n} (-1)^{A(z)} \sum_{y\in\{0,1\}^n} (-1)^{y\cdot z}|y\rangle\end{aligned}$$

The final step in the Grover iteration is to apply W again, to the query register. Also apply W to the query *answer* register, since in the end we will want to make one final query that is not in superposition. Restoring the input and query answer registers, the result is,

$$\frac{2}{2^n} \sum_z (-1)^{A(z)} \frac{1}{2^{n/2}} \sum_y |x\rangle|y\rangle|1\rangle - \frac{1}{2^{n/2}} \sum_z (-1)^{A(z)}|x\rangle|z\rangle|1\rangle$$

Finally, make one more query to A. Since the query answer register contains a 1, we get a negation of $A(z)$ or $A(y)$:

$$\frac{2}{2^n} \sum_z (-1)^{A(z)} \frac{1}{2^{n/2}} \sum_y |x\rangle|y\rangle|\overline{A}(y)\rangle - \frac{1}{2^{n/2}} \sum_z (-1)^{A(z)}|x\rangle|z\rangle|\overline{A}(z)\rangle$$

Call the state that results above $|\Psi\rangle$. Now suppose that $1/4$ of the strings of length n are in A. Then $\sum_z (-1)^{A(z)} = 2^n/2$. In this case,

$$
\begin{aligned}
|\Psi\rangle &= \frac{1}{2^{n/2}}\left(\sum_{y}|x\rangle|y\rangle|\overline{A}(y)\rangle + \sum_{y\in A}|x\rangle|y\rangle|0\rangle - \sum_{y\notin A}|x\rangle|y\rangle|1\rangle\right)\\
&= \frac{2}{2^{n/2}}\sum_{y\in A}|x\rangle|y\rangle|0\rangle.
\end{aligned}
$$

By similar reasoning, if 3/4 of the strings of length n are in A, then

$$|\Psi\rangle = -\frac{2}{2^{n/2}}\sum_{y\notin A}|x\rangle|y\rangle|1\rangle.$$

Consider the observable,

$$\langle\Phi| = \frac{2}{2^{n/2}}\sum_{y}\langle x|\langle y|\langle 0|.$$

It is easy to see that $\langle\Phi \mid \Psi\rangle = 1$ if the number of y's in A is $\frac{1}{4}2^n$, and is 0 if the number of y's in A is $\frac{3}{4}2^n$. This concludes the proof of the lemma. □

Combining Lemmas 3.2 and 3.1, we obtain,

Theorem 3.3 *There is an oracle A such that* $\mathsf{EQP}^A \not\subseteq \mathsf{P}^{\mathsf{NP}^A}$.

4 Discussion and Open Problems

We still do not understand the class EQP very well. The best known upper bound for it, due to Fortnow and Rogers, is that it is contained in LWPP. This implies that it is low for PP, which suggests that it has a low complexity, but its relationship to the polynomial-time hierarchy is very unclear. As pointed out by Berthiaume and Brassard [3], EQP contains the class of languages recognized by nondeterministic Turing machines in which the number of accepting paths is either 0 or 1/2 of all paths; this constitutes the best lower bound known to these authors. To get a better idea of where EQP might lie, it would be nice to establish either that it is in the polynomial-time hierarchy, or that it is not, relative to some oracle. The results of this paper mildly suggest that the latter might be the case, although it should be emphasized that the language that separates EQP^A from $\mathsf{P}^{\mathsf{NP}^A}$ also separates BPP^A from $\mathsf{P}^{\mathsf{NP}^A}$. Separating EQP from the PH will probably require a much more complex test language, and therefore a more sophisticated algorithm for determining membership. Further developments in quantum search algorithms such as that of Grover may offer hints as to how to proceed.

References

[1] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[2] A. Berthiaume. Quantum computation. In L. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, chapter 2, pages 23–50. Springer-Verlag, 1997.

[3] A. Berthiaume and G. Brassard. Oracle quantum computing. In *Journal of Modern Optics*, **41(12)** (1994), pages 2521–2535.

[4] M. Boyer, G. Brassar, P. Høyer, and A. Tapp. Tight bounds on quantum searching. In *Proceedings of the 4th Workshop on Physics and Computation*, (1996), pages 36–43. To appear in *Fortschritte der Physik*. Also see Los Alamos Preprint quant-ph/9605034.

[5] S. Fenner, F. Green, S. Homer and R. Pruim. Determining acceptance pssibility for a quantum computation is hard for the polynomial hierarchy. In *Proceedings of the Royal Society A* **455**, (1999) pages 3953 - 3966.

[6] L. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual Symposium on Theory of Computing,* ACM, 1996, pages 212–219.

[7] P. W. Shor. Polynomial-time algorithms for prime number factorization and discrete logarithms on a quantum computer. *SIAM J. Comp.*, 26:1484–1509, 1997.

[8] L. Stockmeyer. On approximation algorithms for #P. In *SIAM J. Computing* **14** (1985), pages 849-861.