NP-hard problems are not in **BQP**

Reiner Czerwinski
TU Berlin (Alumnus)

Abstract

Grover's algorithm can solve **NP**-complete problems on quantum computers faster than all the known algorithms on classical computers. However, Grover's algorithm still needs exponential time. Due to the BBBV theorem, Grover's algorithm is optimal for searches in the domain of a function, when the function is used as a black box.

We analyze the **NP**-complete set

$$\{(\langle M\rangle, 1^n, 1^t) \mid \text{ TM } M \text{ accepts an } x \in \{0,1\}^n \text{ within } t \text{ steps}\}.$$

If t is large enough, then M accepts each word in $L(M)$ with length n within t steps. So, one can use methods from computability theory to show that black box searching is the fastest way to find a solution. Therefore, Grover's algorithm is optimal for NP-complete problems.

Quantum Complexity Theory Complexity Classes P vs. NP

1 Introduction

One can efficiently simulate a classical computer with a quantum computer so that $\mathbf{P} \subseteq \mathbf{BQP}$. However, there is no efficient quantum algorithm known for **NP**-hard problems.

For any computable function $f : \{0,1\}^n \to \{0,1\}$ with exactly one element x such that $f(x) = 1$, Grover's algorithm [5] can find this element x in $\Theta(2^{n/2})$ accesses to f [8]. There are several variants of Grover's algorithm for the case, that the number of values x with $f(x) = 1$ is not exactly one [1].

Due to the BBBV theorem [3], Grover's algorithm is optimal for searching with a black box. A quantum computer needs to apply at least $\Omega(2^{n/2})$ accesses to the black box.

In this paper, we construct an **NP**-complete problem that we cannot solve faster than with black-box searching. We get an arbitrary TM M and decide, whether there is an input word with size n that would be accepted within t steps. If t is great enough then any input word with size n in $L(M)$ would be accepted within t steps. The number of steps t would grow faster than any computable function. So, we can use methods from computability theory to prove, that we cannot be faster than in the black box manner.

This will infer that every **NP**-hard problem is not in **BQP**.

We take a look to the **NP**-complete set

$$\{(\langle M\rangle, 1^n, 1^t) \mid \text{ TM } M \text{ accepts an } x \in \{0,1\}^n \text{ within } t \text{ steps}\}.$$

The TM M will be fixed, so we denote

$$U_M = \{ \ (1^n, 1^t) \mid \text{ TM } M \text{ accepts an } x \in \{0,1\}^n \text{ within } t \text{ steps}\}.$$

for an arbitrary but fixed M.

If a TM accepts an input, then the TM accepts it within a finite number of steps. So,

$$L(M) = \lim_{t\to\infty} \{x \mid M \text{ accepts } x \text{ within } t \text{ steps}\}$$

.

In section 3, we analyze the set

$$D_M = \{1^n \mid \exists x \in \{0,1\}^n \text{ with } x \in L(M)\}.$$

Obviously, for all $n \in \mathbb{N}$:

$$1^n \in D_M \iff \lim_{t\to\infty} (1^n, 1^t) \in U_M.$$

The set D_M is not computable. But the set is computable relative to the oracle $L(M)$. In this case, we would have to apply the oracle $L(M)$ on each $x \in B^n$, so we need a black box search. We will conclude from the black box complexity of D_M to the complexity of the computable set U_M in section 4. So, the set U_M is in **NP** and not computable faster than with black box search.

2 Notations and Preliminaries

Let M be a Turing machine. We declare $M(x) = 1$ if M accepts the input x. Otherwise, $M(x) = 0$. The language of a Turing machine is the set of accepted words

$$L(M) = \{x \in \{0,1\}^* \mid M(x) = 1\}$$

If M accepts x within at most t steps then $M_t(x) = 1$. So, M_t is a TM for each $t \in \mathbb{N}$ and

$$\lim_{t\to\infty} L(M_t) = L(M) \tag{1}$$

If $n \in \mathbb{N}$ then the unary encoding is defined as

$$1^n = \overbrace{1\ldots1}^{n \text{ times}}$$

In this paper, we use common abbreviations. TM for Turing machine, NTM for a nondeterministic Turing machine, and UTM for a universal Turing machine.

2.1 Rice's Theorem

An index set is a set of TMs with the property: If a TM is in the index set I, then all TMs with the same language are in I, i.e.,

$$\big(\langle M\rangle \in I\big) \wedge \big(L(N) = L(M)\big) \Longrightarrow \langle N\rangle \in I$$

Due to Rice's theorem[7], every non-trivial index set is uncomputable. An index set is trivial, if it is either empty or contains all TMs.

3 c.e. Sets and Quantum Search

We analyze the c.e. set

$$D_M = \{1^n \mid \exists x \in \{0,1\}^n \text{ with } x \in L(M)\} \tag{2}$$

A set is c.e. if it equals a language of a Turing machine. Unfortunately, not every language of a TM is computable. The set D_M is not computable for an arbitrary TM M. But it is computable relative to the oracle $L(M)$.

Theorem 1. *Let M be an arbitrary TM. A quantum computer or OTM with oracle $L(M)$ cannot decide whether $1^n \in D_M$. faster than with a black box search.*

Proof. For $x \in \{0,1\}^n$ the set $\big\{\langle M\rangle \mid \{0,1\}^n \cap L(M) = \{x\}\big\}$ is a non-trivial index set. So, due to Rice's theorem, it is undecidable if an arbitrary but fixed $x \in \{0,1\}^n$ is the only input with length n accepted by M.

So, if one wants to find an $x \in \{0,1\}^n$ with $x \in L(M)$, then one has to apply the oracle $L(M)$ to every $x \in \{0,1\}^n$. $\square$

Due to the BBBV theorem, a quantum computer needs $\Omega(2^{n/2})$ accesses to the oracle to decide whether $1^n \in D_M$ in the worst case.

4 NP versus BQP

The set D_M defined in (2) is not computable. Recall the definition of U_M in the introduction:

$$U_M = \{(1^n, 1^t) \mid \exists x \in \{0,1\}^n \text{ with } x \in L(M_t)\} \tag{3}$$

The set U_M is similar to D_M, except it is computable. The set $\{0,1\}^n$ is finite. So, equation (1) implies

$$1^n \in D_M \iff \exists t \in \mathbb{N} : (1^n, 1^t) \in U_M \tag{4}$$

Lemma 1. *For any TM M the set U_M is in **NP**.*

Proof. One wants to check whether $(1^n, 1^t) \in U_M$.

An NTM can choose an $x \in \{0,1\}^n$. After that, it can simulate the calculation of M for t steps. The run time of the NTM is $n + t$, which equals the length of the element $(1^n, 1^t)$. $\square$

For the set D_M defined in (2) we have proven that a computer, even a quantum computer, cannot search faster than in the black box manner even with an oracle for $L(M)$. Now we will conclude this result to the set U_M.

Theorem 2. *Let $n, t \in \mathbb{N}$ be arbitrary but fixed and M an arbitrary TM.Testing whether $(1^n, 1^t) \in U_M$ cannot be done faster than with a black box search.*

Proof. For a fixed n, there is a $t \in \mathbb{N}$, such that $1^n \in D_M \iff (1^n, 1^t) \in U_M$. See equation (4). In this case for all $x \in \{0,1\}^n$: $x \in L(M_t) \iff x \in L(M)$.

Instead of the oracle $L(M)$, one can use the computable oracle $L(M_t)$. For a sufficiently large t, it does not matter if one uses $L(M)$ or $L(M_t)$. So, testing whether $(1^n, 1^t) \in U_M$ with oracle $L(M_t)$ is as fast as testing whether $1^n \in D_M$ with oracle $L(M)$. Due to theorem 1 one cannot test it faster than with the black box search.

Obviously, a TM or quantum computer without an oracle for $L(M_t)$ is not faster than a machine with this oracle.

Maybe there is an $x \in \{0,1\}^n$ with $x \in L(M)$, but $x \notin L(M_t)$. Assume, that one could test whether $(1^n, 1^t) \in U_M$ faster than with black box search in this case. Then one could decide if there is such an x. But this is undecidable. So, one needs a black box search for any t when the TM M is arbitrary. $\square$

Theorem 2 and the BBBV theorem imply that a quantum computer has the run time of $\Omega(2^{n/2}$ to decide whether $(1^n, 1^t) \in U_M$ for U_M defined in (3) in the worst case. This implies the main results:

Corollary 1. $\boldsymbol{NP} \nsubseteq \boldsymbol{BQP}$

Proof. For an arbitrary TM M, the set U_M described in (3) is in **NP** but not in **BQP**. $\square$

Corollary 2. ***BQP** does not include **NP**-hard problems*

Proof. Let X be an **NP**-hard set. Assume that $X \in$ **BQP**. Then there is a reduction $U_M \leq_T^P X$ for the set U_M described in (3) with an arbitrary TM M. So $U_M \in$ **BQP** for any TM M, which is refuted by theorem 1. $\square$

Corollary 3. $\boldsymbol{P} \neq \boldsymbol{NP}$

Proof. Assume **P** = **NP**. Thus, **NP** $\subseteq$ **BQP** because of **P** $\subseteq$ **BQP**. But **NP** $\subseteq$ **BQP** is refuted by corollary 1. $\square$

5 Conclusion

The result of this paper includes the solution to the famous **P** vs. **NP** problem. As Oded Goldreich mentioned on his web page [4], one needs a novel insight to solve this problem. Someone anonymous said: “Experts agree that one cannot solve problems like **P** vs. **NP** with simple computability tricks.” This might be the reason why no one had tried it before.

Why does this proof method circumvent the relativization barrier? There are oracles with **P** = **NP** relative to them. But with such an oracle one can test many inputs simultaneously. A common TM cannot do this.

Acknowledgments

I acknowledge Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani for the remarkable BBBV theorem. This theorem shortened this paper.

References

[1] Andris Ambainis. Quantum search algorithms. *ACM SIGACT News*, 35(2):22–35, 2004.

[2] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[3] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[4] O. Goldreich. On resolving the p vs np problem. `https://www.wisdom.weizmann.ac.il/~oded/p-vs-np.html`. Accessed: 2023-02-09.

[5] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[6] Matthias Homeister. *Quantum Computing verstehen*. Springer, 2008.

[7] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953.

[8] Yaoyun Shi. Quantum and classical tradeoffs. *Theoretical computer science*, 344(2-3):335–345, 2005.