Quantum Computing in Complexity Theory and Theory of Computation

Austin Mohr
Southern Illinois University at Carbondale
Carbondale, IL 62901
E-mail: austinmohr@gmail.com

Abstract

Traditionally, complexity theory has concerned itself with algorithms run by classical computers. With the recent developments in quantum computing, complexity theorists have begun considering just how quantum algorithms fit into the picture. After a brief introduction into classical complexity theory, we consider what is known and what is speculated about the relationship of the class of problems solvable by quantum algorithms (denoted BQP) and the well-known classical classes P and NP. A short discussion on the possibility of hypercomputation (that is, the ability to solve problems which are undecidable on a classical Turing machine) follows.

1 Introduction

Computational complexity theory (or just "complexity theory") is the study of the scalability of algorithms, both in general and in a problem-specific sense. The term "scalability" refers to how the time and/or space required to solve a problem grows as the input grows. The notion of growth is formalized through the use of big oh notion.

Definition 1.1 (Big Oh [2])**.** Suppose that a_n converges to A, b_n converges to 0, with $a_n \neq 0$ and $b_n \neq 0$ for any $n \in N$. Then a_n converges to A with a rate of convergence $O(b_n)$ if and only if $|a_n - A| \leq K|b_n|$ for sufficiently large values of n and some positive constant K.

In this way, one can consider the scalability of an algorithm without the added considerations of processor speed, implementation language, machine architecture, and so on. An algorithm which is $O(n^3)$ is said to be "harder" than one which is $O(n^2)$ because the former will require, in general, more operations to complete than the latter, regardless of the speed at which these operations are performed. A problem is said to be solvable in polynomial time if there exists an algorithm whose runtime can be expressed as $O(n^p)$, $p \in \mathbb{N}$. Otherwise, the problem is said to be non-polynomial.

The study of complexity theory allows one to place practical limits on what can be accomplished by a computer. The emergence of quantum algorithms has required complexity theorists to reconsider the relationship of existing problem classes and even invent new ones. The following sections will introduce some key concepts from both complexity theory and the theory of computation and discuss the potential ramifications of quantum computing on these fields.

2 Complexity Theory

The fundamental unit of complexity theory is the complexity class. A given complexity class consists of problems which all possess some similar characteristic regarding their hardness. By far, the two most important complexity classes are P and NP. P is defined to be the class of problems for which there is an efficient (i.e. polynomial time) deterministic algorithm. NP is defined to be the class of problems for which there is an efficient deterministic certification algorithm. That is, regardless of how difficult the problem is to actually solve, a problem is in NP if we can verify that a proposed solution is indeed a true solution to the problem in polynomial time. [1] We can make the following observation.

Proposition 2.1. P $\subseteq$ NP

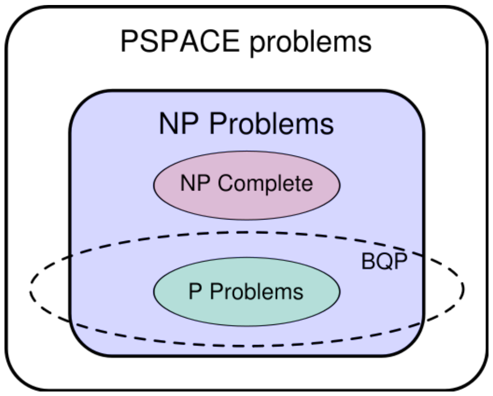
Proof. Let a problem x be in P. Then, there exists a deterministic polynomial time algorithm for solving x which will produce an output y. Now, suppose we are given y and wish to efficiently certify that it is indeed a solution. We will simply run the polynomial time algorithm and check that its output is indeed y. Hence, we have a polynomial time certification algorithm for x. Therefore, x is in NP. □

Perhaps the most famous open question in computer science is P $\overset{?}{=}$ NP. While it is generally believed that P $\neq$ NP, there is no proof of this fact. The problem is so fundamental that the Clay Mathematics Institute included it in its list of seven *Millenium Prize Problems*. A correct proof to any of these seven problems will make the prover $1 million richer.

A useful tool in classifying problems into one class or another is the notion of a polynomial reduction. Informally, problem A is polynomially reducible to problem B if an algorithm for B can be called a polynomial number of times to construct a solution for A. Even more informally, one could say simply that B is harder than A. A problem is said to belong to the class NP-Complete if every problem in NP can be polynomially reduced to this problem. So, NP-Complete problems are the hardest problems in NP. NP-Complete problems are of particular importance because an efficient solution to even a single such problem can be used to efficiently solve all problems in NP.

3 Quantum Computing in Complexity Theory

The class of problems efficiently solvable by a quantum computer is known as Bounded-Error Quantum Polynomial Time (BQP). It's classical counterpart is called BPP. Due to the non-deterministic nature of quantum mechanics, a quantum computer can only run probabalistic algorithms. Hence, the algorithms in BQP are written in such a way that they fail only a negligably small amount of the time (hence, "bounded-error"). In general, probabilistic algorithms (both classical and quantum) provide considerable decreases in runtime. However, little is known regarding the relationship of P, NP, BPP, and BQP. The following figure summarizes the belief held by most complexity theorists. Note that BPP is omitted, but is known to be a superset of P and is suspected to be a subset of BQP.



Suspected (though unproven) relationship between P, NP, and BQP [6]

Note that, to the chagrin of quantum computer scientists, BQP is suspected to be disjoint from NP-Complete. If this is true, then many of the truly hard problems are still not solvable in polynomial time, even by a quantum computer. That is not to say, however, that the algorithms in BQP are useless. Often, quantum algorithms are faster than their classical counterparts because they are able to maintain a superposition of all states of a particular system, and then select a particular state from among the list with just a single operation. To accomplish the same task requires $O(n)$ operations on a classical computer. This advantage has allowed Grover to reduce the time spent searching an unsorted database from $O(n)$ to $O(n^{1/2})$. Similarly, Shor's algorithm takes integer factorization from the superpolynomial runtime $O(e^{c(logn)^{1/3}(loglogn)^{2/3}})$ to $O(n^3)$ which is polynomial (and much easier to write). However, how and whether BQP interacts with NP-Complete is still an open problem.

4 Hypercomputation

Definition 4.1. A function is said to be Turing-computable if there exists a finite procedure (called an algorithm) for computing the output of the function.

When discussing complexity theory, it is understood that the algorithms in question are Turing-computable. That is, the the algorithm will halt after a finite number of steps. The so-called Church-Turing thesis states that all functions which are "naturally regarded as computable" are Turing-computable. This seems like a fair (and rather obvious) assumption, given the broad definition of Turing-computability. However, the concept of hypercomputation has become a hot research topic in light of the advancements in quantum computation. A hypercomputer is a machine that can solve problems which are not Turing-computable (such as the Halting Problem). Unfortunately, it has been shown in [3] that the current quantum computational model is Turing-reducible. That is, a quantum computer may be faster than a classical one, but it cannot solve any problems that a classical computer cannot. At the same time, [4] presents theoretical models for hypercomputation such as Oracle-Machines (or just O-Machines) and Infinite State Turing Machines. Ord holds that these models may be physically realizable, but require a much more powerful form of quantum superposition than is currently understood, such as infinite state superposition.

5 Conclusion

In conclusion, while quantum computing offers incredible increases in speed compared to classical computing, whether this new paradigm offers anything truly new is yet to be seen. Indeed, it is widely believed (though unproven) that the any problem solvable by the current model of quantum computing is also solvable by a classical Turing machine. The prospect of hypercomputation is intriguing, though the possibility of realizing such a machine is still only theoretical. However, the increased power of quantum computers means that a quantum computer can always solve a given problem instance faster than a classical computer. This alone is reason enough to continue research into this burgeoning field.

References

[1] Kleinberg, Jon and Eva Tardos. *Algorithm Design*. Boston: Pearson Education, 2006.

[2] Kosmala, Witold A. J. *A Friendly Introduction to Analysis*. New Jersey: Pearson Prentice Hall, 2004.

[3] Michael Nielsen and Isaac Chuang (2000). *Quantum Computation and Quantum Information*. Cambridge: Cambridge University Press.

[4] Ord, Toby. “Hypercomputation: computing more than a Turing machine”. (http://arxiv.org/ftp/math/papers/0209/0209332.pdf). University of Melbourne.

[5] “Computational complexity theory.” Wikipedia, The Free Encyclopedia. 25 Apr 2007, 13:21 UTC. Wikimedia Foundation, Inc. 3 May 2007 (http://en.wikipedia.org/w/index.php?title=Computational_complexity_theory&oldid=125824877).

[6] “Quantum computer.” Wikipedia, The Free Encyclopedia. 2 May 2007, 19:50 UTC. Wikimedia Foundation, Inc. 4 May 2007 (http://en.wikipedia.org/w/index.php?title=Quantum_computer&oldid=127764104).