P≠NP - A Definitive Proof by Contradiction

Adnan Masood

April 1, 2014

Abstract

The P versus NP problem *use to be* a major unsolved problem in computer science. However, as of today, April 1st 2014, the researcher has proven sans rigor that P≠NP.The question of $P = NP$ asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer. This problem was introduced in 1971 by Stephen Cook in his seminal paper "The complexity of theorem proving procedures" Beame et al. [1995], Cook [2000] and is considered by many to be the most important open problem in the field. In this proof, we reduce this important concept to an algebraic proof *reductio ad absurdum*. Algebrizing proofs has gotten a bad reputation since Baker-Gill-Solovay result, however new non-relativizing proof techniques were successfully used to prove that $IP = PSPACE$. Following the Erik Demaine's non-proof of $P = NP$[1], Scott Aaronson and Avi Wigderson also showed that the main technical tool used in the IP = PSPACE proof, known as arithmetization, was also insufficient to resolve P = NP.

Trivializing the solution to the $P = NP$ question, this paper deterministically determines that $P \neq NP$ and that the NP problems can be verified in polynomial time; like the subset-sum problem, can also be solved in polynomial time. As it proves $P \neq NP$, it means that there are problems in NP (such as NP-complete problems) that are harder to compute than to verify: they could not be solved in polynomial time, but the answer could be verified in polynomial time. This proof gives NSA much needed relief from bitcoin mining as well as public-key cryptography and symmetric ciphers vulnerabilities using Qubit Miner ASIC and further focus on digital eavesdropping.

1 Introduction

To start with a circular definition, P versus NP problem is the determination of whether all NP-problems are actually P-problems. If P and NP are not equivalent, then the solution of NP-problems requires (in the worst case) an exhaustive search, while if they are, then asymptotically faster algorithms may exist. A problem is assigned to the NP (non-deterministic polynomial time) class if it is solvable in polynomial time by a non-deterministic Turing machine. A P-problem (whose solution time is bounded by a polynomial) is always also NP. If a problem is known to be NP, and a solution to the problem is somehow known, then demonstrating the correctness of the solution can always be reduced to a single P (polynomial time) verification. If P and NP are not equivalent, then the solution of NP-problems requires (in the worst case) an exhaustive search.Weisstein

Now that we have formalities out of the way, a prior Hubert Chen's argument that "P-not-equal-to-NP": " proof by contradiction is in order. *It is assumed $P = NP$. Let y be a proof that $P = NP$. The proof y can be verified in polynomial time by a competent computer scientist, the existence of which we assert. However, since P=NP, the proof y can be generated in polynomial time by such computer scientists. Since this generation has not yet occurred (despite attempts by such computer scientists to produce a proof), we*

[1] https://www.youtube.com/watch?v=VqeF98GGiXQ

have a contradiction. We however feel that mere onus proband is insufficient and without proper use of *argumentum ad ignorantiam, cum hoc ergo propter hoc* and *reductio ad absurdum*, this proof won't stand.

Following proof that showed that $P \neq NP$ lacks the practical computational benefits of a proof that $P = NP$, but nevertheless represents a very significant advance in computational complexity theory and provide guidance for future research. It allows one to show in a formal way that many common problems cannot be solved efficiently, so that the attention of researchers, bitcoin enthusiasts, and NSA can be focused on partial solutions or solutions to other problems.

1.1 Advantages of the Proof $P \neq NP$

This proof of $P \neq NP$ leaves open the average-case complexity of hard problems in NPJohnson [2012]. For example, it is possible that SAT requires exponential time in the worst case, but that almost all randomly selected instances of it are efficiently solvable. A proof that P = NP would have had stunning practical consequences, and therefore is avoided. If the proof leads to efficient methods for solving some of the important problems in NP, doctoral students are better served by avoiding it. It is also possible that a proof may not lead directly to efficient methods, perhaps if the proof is non-constructive, or the size of the bounding polynomial is too big to be efficient in practice.

Cryptography, for example, relies on certain problems being difficult. A constructive and efficient solution to an NP-complete problem such as 3-SAT would break most existing cryptosystems including public-key cryptography and symmetric ciphers such as AES or 3DES used for the encryption of communications data. These would need to be modified or replaced by information-theoretically secure solutions which would requite lots of work. Therefore, we strongly belief that $P \neq NP$ serves graduate students (humanity) quite well in general.

To a graduate student's delight, this also helps to successfully avoid the wide range of satisfiability problems (often referred to as SAT) 0-1 integer programming Clique, set packing Vertex cover Set covering, Feedback arc set, Directed Hamilton circuit, Undirected Hamilton circuit, Satisfiability with at most 3 literals per clause, Chromatic number (also called the Graph Coloring Problem), Clique cover, Exact cover, Knapsack, Job sequencing Partition and Max cut to name a few (too many if you ask me). Godel, in his early thoughts on computational complexity, noted that a mechanical method that could solve any problem would revolutionize mathematics; but knowing Godel, he must have left it incomplete.

2 The Proof

Research mathematicians spend their careers trying to prove theorems, and some proofs have taken decades or even centuries to find after problems have been stated—for instance, Fermat's Last Theorem took over three centuries to prove. A method that is guaranteed to find proofs to theorems, should one exist of a "reasonable" size, would essentially end this struggle. Here is a simple, algebraic proof by contradiction.

In logic, proof by contradiction is a form of proof that establishes the truth or validity of a proposition by showing that the proposition's being false would imply a contradiction. Proof by contradiction is also known as indirect proof, apagogical argument, proof by assuming the opposite, and reductio ad impossibilem. It is a particular kind of the more general form of argument known as reductio ad absurdum.

Let's assume $p = \mathscr{N}\mathscr{P}$

$$p^2 = p\mathscr{N}\mathscr{P}$$

$$p^2 + p^2 = p\mathscr{N}\mathscr{P} + p^2$$

$$2p^2 = p\mathscr{N}\mathscr{P} + p^2$$

$$2p^2 - 2p\mathscr{N}\mathscr{P} = p\mathscr{N}\mathscr{P} + p^2 - 2p\mathscr{N}\mathscr{P}$$

$$2p^2 - 2p\mathscr{N}\mathscr{P} = p^2 - p\mathscr{N}\mathscr{P}$$

$$2(p^2 - p\mathscr{N}\mathscr{P}) = 1(p^2 - p\mathscr{N}\mathscr{P})$$

Canceling $(p^2 - p\mathscr{N}\mathscr{P})$ on both sides, we get the contradiction $2 = 1$, which is incorrect, therefore

$$p \neq \mathscr{N}\mathscr{P}$$

3 Conclusion

$P = NP$ asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer. It was introduced in 1971 by Stephen Cook in his seminal paper "The complexity of theorem proving procedures" and is considered by many to be the most important open problem in the field. In this proof, we reduce this important concept to an algebraic proof *reductio ad absurdum* .

P=NP is one of the seven Millennium Prize Problems selected by the Clay Mathematics Institute to carry a US$1,000,000 prize for the first correct solution. The millennium prize committee can reach the author at adnan @ nova dot edu. Researcher does not accept bitcoins.

References

Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of np search problems. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 303–314. ACM, 1995.

Stephen Cook. The p versus np problem. In *Clay Mathematical Institute; The Millennium Prize Problem*. Citeseer, 2000.

David S Johnson. A brief history of np-completeness, 1954–2012. *Documenta Mathematica*, pages 359–376, 2012.

Eric W. Weisstein. Tree. From MathWorld—A Wolfram Web Resource. URL `http://mathworld.wolfram.com/Tree.html`. Last visited on 13/4/2012.