P Vs NP Problem and its Application in Public Key Cryptography

Aashirvad Mohanty
HMRITM, ECE Deptt.
Hamidpur, Delhi, India.

Abhishek Gahlawat
HMRITM, CSE Deptt.
Hamidpur, Delhi, India.

Abstract: **This paper deals with the latest development that have took place to solve P vs NP problem. We will look into different ways which have tried to give a solution to this problem. With a deep and thorough analysis of various works by different authors around the world, it can be concluded that the problem is still unresolved but there is a lot of scope still left to explore which requires further research.**

Keywords:- Public key cryptography, circuit complexity, harder problems.

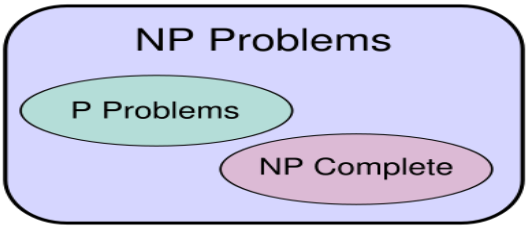
1. INTRODUCTION

The P versus NP problem is a major unsolved problem in computer science and is considered by many to be the most important open problem in the field. It is one of the seven Mil-lennium Prize Problems selected by the Clay Mathematics Institute to carry a US$1,000,000 for the first correct proof. In-formally, it asks whether every problem whose solution can be quickly verified by a computer, can also be quickly solved by a computer. According to Computational complexity theory, the class P consists of all those decision problems that can be solved on a deterministic sequential machine in an amount of time that is polynomial in the size of the input; the class NP consists of all those decision problems whose positive solu-tions can be verified in polynomial time given the right infor-mation, or equivalently, whose solution can be found in poly-nomial time on a non-deterministic machine.

2. HISTORY

Although the **P** versus **NP** problem was formally defined in 1971, there were previous inklings of the problems involved, the difficulty of proof, and the potential consequences. In 1955, mathematician John Nash wrote a letter to the NSA, where he speculated that cracking a sufficiently complex code would require time exponential in the length of the key.If proved (and Nash was suitably skeptical) this would imply what we today would call **P** ≠ **NP**, since a proposed key can easily be verified in polynomial time. Another mention of the underlying problem occurred in a 1956 letter written by Kurt Gödel to John von Neumann. Gödel asked whether theorem-proving (now known to be co-NP-complete) could be solved in quadratic or linear time,and pointed out one of the most important consequences - that if so, then the discovery of mathematical proofs could be automated.

3. P VERSUS NP PROBLEM?



Suppose we have a large group of students that we need to pair up to work on projects. We know which students are compatible with each other and we want to put them in compatible groups of two. We could search all possible pairings but even for 40 students we would have more than three hundred billion trillion possible pairings.
But many related problems do not have such an efficient algorithm.What if we put the students into three groups so that each student is in the same group with only his or her compatibles (3-Coloring)? So P = NP means that for every problem that has an efficiently verifiable solution, we can find that solution efficiently as well.
We call the very hardest NP problems which includes:
1.Partition Into Triangles
2.Clique
3.Hamiltonian Cycle
4.-Coloring

4.NP-COMPLETENESS

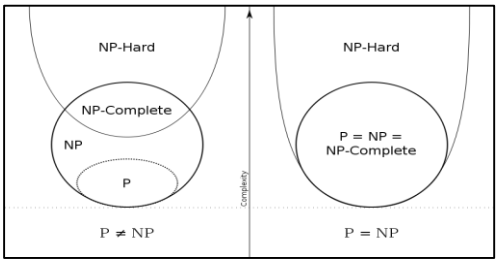


Fig2:Euler diagram for P, NP, NP-complete, and NP-hard set of problems.

To attack the **P** = **NP** question, the concept of **NP**-completeness is very useful. **NP**-complete problems are a set of problems to each of which any other **NP**-problem can be reduced in polynomial time, and whose solution may still be verified in polynomial time. That is, any **NP** problem can be transformed into any of the **NP**-complete problems. Informally, an **NP**-complete

problem is an **NP** problem that is at least as "tough" as any other problem in **NP**.[1]

Based on the definition alone it is not obvious that **NP**-complete problems exist; however, a trivial and contrived **NP**-complete problem can be formulated as follows: given a description of a Turing machine M guaranteed to halt in polynomial time, does there exist a polynomial-size input that M will accept? It is in **NP** because (given an input) it is simple to check whether M accepts the input by simulating M; it is **NP**-complete because the verifier for any particular instance of a problem in **NP** can be encoded as a polynomial-time machine M that takes the solution to be verified as input. Then the question of whether the instance is a yes or no instance is determined by whether a valid input exists.[2]

The first natural problem proven to be **NP**-complete was the Boolean satisfiability problem. As noted above, this is the Cook–Levin theorem; its proof that satisfiability is **NP**-complete contains technical details about Turing machines as they relate to the definition of **NP**. However, after this problem was proved to be **NP**-complete, proof by reduction provided a simpler way to show that many other problems are also **NP**-complete, including the subset sum problem discussed earlier. Thus, a vast class of seemingly unrelated problems are all reducible to one another, and are in a sense "the same problem".

5.WHAT IF P=NP?

To understand the importance of the P versus NP problem let us imagine a world where P = NP. Technically we could have P = NP but not have practical algorithms for most NP-complete problems. But suppose in fact that we do have very quick algorithms for all these problems.

P = NP would also have big implications in mathematics. One could find short fully logical proofs for theorems but these fully logical proofs are usually extremely long. But we can use the Occam razor principle to recognize and verify mathematical proofs as typically written in journals. We can then find proofs of theorems that have reasonably length proofs say in under 100 pages. A person who proves P = NP would walk home from the Clay Institute not with one million-dollar check but with seven (actually six since the Poincare Conjecture appears solved).

6.CIRCUIT COMPLEXITY

To show P 6= NP it is su cient to show some NPcomplete problem cannot be solved by relatively small circuits of AND, OR and NOT gates (the number of gates bounded by a fixed polynomial in the input size). Saxe and Sipser [16] showed that small circuits cannot solve the parity function if the circuits have a fixed number of layers of gates. In 1985, Razborov showed the NPcomplete[7] problem of nding a large clique does not have small circuits if one only allows AND and OR gates (no NOT gates). If one extends Razborov's result to general circuits one will have proved P 6= NP. Razborov later showed his techniques would fail miserably if one allows NOT gates Razborov and Rudich develop a notion of \natural" proofs and give evidence that our limited techniques in circuit complexity cannot be pushed much further. And in fact we haven't seen any significantly new circuit lower bounds in the past twenty years.

7.HARDER PROBLEMS

Although it is unknown whether **P** = **NP**, problems outside of **P** are known. A number of succinct problems (problems that operate not on normal input, but on a computational description of the input) are known to be EXPTIME-complete. Because it can be shown that **P** ≠ EXPTIME(experiment time) these problems are outside **P**, and so require more than polynomial time. Examples include finding a perfect strategy for chess (on an $N \times N$ board) and some other board games.

Even more difficult are the undecidable problems, such as the halting problem. They cannot be completely solved by any algorithm, in the sense that for any particular algorithm there is at least one input for which that algorithm will not produce the right answer; it will either produce the wrong answer, finish without giving a conclusive answer, or otherwise run forever without producing any answer at all.

8.USING HARDNESS

we saw the nice world that arises when we assume P = NP. But we expect P ≠NP to hold in very strong ways. We can use strong hardness assumptions as a positive tool, particularly to create cryptographic protocols and to reduce or even eliminate the need of random bits in probabilistic algorithms.[21]

9.NP PROBLEM: NEITHER P NOR NP COMPLETE

It was shown by Ladner that if **P** ≠ **NP** then there exist problems in **NP** that are neither in **P** nor **NP**-complete. Such problems are called **NP**-intermediate problems.[8] The graph isomorphism problem, the discrete logarithm problem and the integer factorization problem are examples of problems believed to be **NP**-intermediate. They are some of the very few **NP** problems not known to be in **P** or to be **NP**-complete.The graph isomorphism problem is the computational problem of determining whether two finite graphs are isomorphic. An important unsolved problem in complexity theory is whether the graph isomorphism problem is in **P**, **NP**-complete, or **NP**-intermediate. The answer is not known, but it is believed that the problem is at least not **NP**-complete. If graph isomorphism is **NP**-complete, the polynomial time hierarchy collapses to its second level. Since it is widely believed that the polynomial hierarchy does not collapse to any finite level, it is believed that graph isomorphism is not **NP**-complete. The best algorithm for this problem, due to Laszlo Babai and Eugene Luks, has run time $2^{O(\sqrt{n}\log(n))}$ for graphs with n vertices.

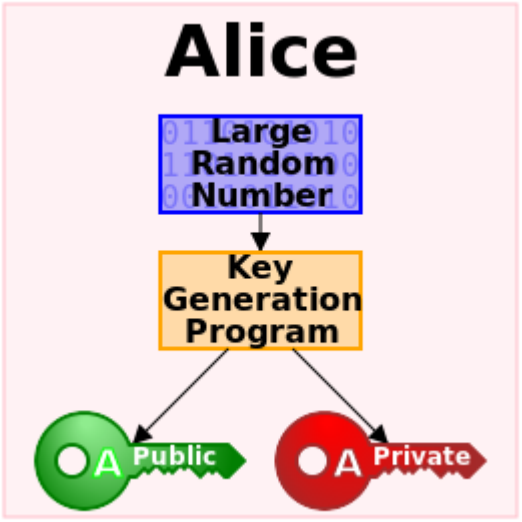
10. APPROXIMATION

We cannot hope to solve NP-complete optimization problems exactly but often we can get a good approximate answer. Consider the traveling salesperson problem again with
distances between cities given as the crow ies (Euclideandistance). This problem remains NP-complete but Arora gives an efficient algorithm that gets very close to the best
possible route. Consider the MAX-CUT problem of dividing people into two groups to maximize the number of incompatibles between the groups. Goemans and Williamson [18] uses semi-definite programming to give a division of people only a .878567 factor of the best possible.

11.INTERACTIVE PROOFS AND LIMITS OF APPROXIMATION

Many times though we seem to hit a limit on our ability to even get good approximations. We now know that we cannot achieve better approximations on many of these problems
unless P = NP and we could solve these problems exactly. The techniques to show these negative results came out of a new model of proof system originally developed for cryptog-
raphy and to classify group theoretic algorithmic problems.

12. APPLICATION:PUBLIC KEY CRYPTOGRAPHY



Public key cryptography, or **asymmetric cryptography**, is any cryptographic system that uses pairs of keys: *public keys* which may be disseminated widely, and *private keys* which are known only to the owner. This accomplishes two functions: authentication, which is when the public key is used to verify that a holder of the paired private key sent the message, and encryption, whereby only the holder of the paired private key can decrypt the message encrypted with the public key.
In a public key encryption system, any person can encrypt a message using the public key of the receiver, but such a message can be decrypted only with the receiver's private key. For this to work it must be computationally easy for a user to generate a public and private key-pair to be used for encryption and decryption. The strength of a public key cryptography system relies on the degree of difficulty (computational impracticality) for a properly generated private key to be determined from its corresponding public key. Security then depends only on keeping the private key private, and the public key may be published without compromising security.

If P = NP then public-key cryptography is impossible. Assuming P ≠ NP is not enough to get public-key protocols, instead we need strong average-case assumptions about the difficulty of factoring or related problems. We can do much more than just public-key cryptography using hard problemsOn-line poker is generally played through some trusted website, usually somewhere in the Caribbean. Can
we play poker over the Internet without a trusted server? Using the right cryptographic assumptions, not only poker but any protocol that uses a trusted party can be replaced by one that uses no trusted party and the players can't cheat or learn anything new beyond what they could do with the trusted party above.

13. ELIMINATING RANDOMNESS

Most notably there were probabilistic algorithms for determining whether a number is prime. Truly independent and uniform random bits are either very difficult or impossible to produce. Computer algorithms instead use pseudorandom generators to generate a sequence of bits from some given seed. The generators typically found on our computers usually work well but occasionally give incorrect results both in theory and in practice. We can create theoretically better pseudorandom genera-tors in two different ways, one based on the strong hardness assumptions of cryptography and the other based on worstcase complexity assumptions. I will focus on this second approach.

14.COULD QUANTUM COMPUTERS SOLVE NPCOMPLETE PROBLEMS?

While we have randomized and non-randomized efficient algorithms for determining whether a number is prime, these algorithms usually don't give us the factors of a composite number. Much of modern cryptography relies on the fact that factoring or similar problems do not have efficient algorithms. So could quantum computers one day solve NP-complete problems? Unlikely. Lov Grover did find a quantum algorithm that works on general NP problems but that algorithm only achieves a quadratic speed-up and we have evidence that those techniques will not go further. Meanwhile quantum cryptography, using quantum mechanics to achieve some cryptographic protocols without

hardness assumptions, has had some success both in theory and in practice.

15. CONCLUSION

The P versus NP problem has gone from an interesting problem related to logic to perhaps the most fundamental and important mathematical question of our time, whose importance only grows as computers become more powerful and widespread. Proving P≠NP would not be the end of the story, it would just show that NP-complete problem don't have efficient algorithms for all inputs but many questions might remain. Cryptography for example would require that a
problem like factoring (not believed to be NP-complete) is hard for randomly drawn composite numbers. Proving P 6= NP might not be the start of the story either. Weaker separations remain perplexingly di cult, for example showing that Boolean-formula Satisfiability cannot be solved in near-linear time or showing that some
problem using a certain amount of memory cannot be solved using roughly the same amount of time.
None of us truly understand the P versus NP problem, we have only begun to peel the layers around this increasingly complex question. Perhaps we will see a resolution of the P versus NP problem in the near future but I almost hope not. The P versus NP problem continues to
inspire and boggle the mind and continued exploration of this problem will lead us to yet even new complexities in that truly mysterious process we call computation.

ACKNOWLEDGMENTS

Thanks to Assistant Prof. Kuldeep Kumar for many useful discussions and comments. I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the study. Last but not the least, I acknowledge my friends for their contribution in the completion of the research work.

REFERENCES

[1] The international SAT competitions web page. http://www.satcompetition.org.
[2] S. Aaronson. Is P versus NP formally independent? Bulletin of the European Association for Theoretical Computer Science, 81, Oct. 2003.
[3] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. Annals of Mathematics, 160(2):781{793, 2004.
[4] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. On the solution of traveling salesman problems.
[5] DOCUMENTA MATHEMATICA, Extra Volume ICM III:645{656, 1998.
[6] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. Journal of the ACM, 45(5):753{782, Sept. 1998.
[7] S. Arora and B. Barak. Complexity Theory: A Modern Approach. Cambridge University Press, Cambridge, 2009.
[8] T. Baker, J. Gill, and R. Solovay. Relativizations of the P = NP question. SIAM Journal on Computing, 4(4):431{442, 1975.
[9] B. Cipra. This ising model is NP-complete. SIAM News, 33(6), July/August 2000.
[10] V. Conitzer and T. Sandholm. New complexity results about Nash equilibria. Games and Economic Behavior, 63(2):621{641, July 2008.
[11] S. Cook. The complexity of theorem-proving procedures. In Proceedings of the 3rd ACM Symposium on the Theory of Computing, pages 151{158. ACM, New York, 1971.
[12] R. Downey and M. Fellows. Parameterized Complexity. Springer, 1999.
[13] J. Edmonds. Paths, trees and owers. Canadian Journal of Mathematics, 17:449{467, 1965.
[14] L. Fortnow and W. Gasarch. Computational complexity. http://weblog.fortnow.com.
[15] L. Fortnow and S. Homer. A short history of computational complexity. Bulletin of the European Association for Theoretical Computer Science, 80, June 2003. Computational Complexity Column. complexity of some boolean functions. Soviet Mathematics{Doklady, 31:485{493, 1985.
[16] A. Razborov. On the method of approximations. In Proceedings of the 21st ACM Symposium on the Theory of Computing, pages 167{176. ACM, New York, 1989.
[17] A. Razborov and S. Rudich. Natural proofs. Journal of Computer and System Sciences, 55(1):24{35, Aug. 1997.
[18] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5):1484-1509, 1997.
[19] R. Solovay and V. Strassen. A fast Monte-Carlo test for primality. SIAM Journal on Computing, 6:84{85, 1977. See also erratum 7:118, 1978.
[20] M. Sudan. Probabilistically checkable proofs. Communications of the ACM, 52(3):76{84, Mar. 2009.
[21] R. Trakhtenbrot. A survey of Russian approaches to Perebor (bruteforce search) algorithms. Annals of the History of Computing, 6(4):384{400, 1984.
[22] A. Turing. On computable numbers, with an application to the Etscheidungs problem. Proceedings of the London Mathematical Society, 42:230{265, 1936.
[23] D. van Melkebeek. A survey of lower bounds for satis ability and related problems. Foundations and Trends in Theoretical Computer Science, 2(197-303), 2007.