ISSN:0975-9646

P VS NP PROBLEM

Shivam Sharma

Shivam Sharma has completed his B.Tech, Computer Science and Engineering from JSS Academy of Technology, Noida

ABSTRACT: **This paper deals with the most recent 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. The paper includes the proof complexity and various other aspects which enlightens the research that have taken place in the field so far. 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: **Clique, cloning, brute force.**

1. Introduction

As we solve larger and more complex problems with greater computational power and cleverer algorithms, the problems we cannot tackle begin to stand out. The theory of NP-completeness helps us understand these limitations and the P versus NP problems begins to loom large not just as an interesting theoretical question in computer science, but as a basic principle that permeates all the sciences. So while we don't expect the P versus NP problem to be resolved in the near future, the question has driven researchin a number of topics to help us understand, handle and even take advantage of the hardness of various computational problems.

In this survey we will look at how various researches have tried to solve the P versus NP problem but also how this question has shaped so much of the research in computer science and be-yond. We will look at how to handle NP-complete problems and the theory that has developed from those approaches. We show how a new type of interactive proof systems led to limitations of approximation algorithms. We consider whether quantum computer can solve NP-complete problems. We end describing a new long-term project that will try to separate P from NP using algebraic-geometric techniques.

2. What Is The P Versus Np Problem?

Suppose there is a large group of students that we need to pair up to work on a presentation. We have the knowledge which students are compatible with each other and we want to put them in compatible groups of two. Looking at all possible pairs, 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 wanted to make groups of three students with each pair of students in each group compatible (Partition Into Triangles)? What if we wanted to find a large group of students all of whom are compatible with each other (Clique)? What if we wanted to sit the students around a large round table with no incompatible students sitting next to each other (Hamiltonian Cycle)? 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:

- Partition Into Triangles
- Clique
- Hamiltonian Cycle
- 3-Coloring

NP-complete, i.e. given an efficient algorithm for one of them, we can find an efficient algorithm for all of them and in fact any problem in NP.

As computers grew cheaper and more powerful, computation started playing a major role in nearly every academic field, especially the sciences. The more scientists can do with computers, the more they realize some problems seem computationally difficult. Many of these fundamental problems turn out to be NP-complete.

3. 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.

Many focus on the negative, that if P = NP then public-key cryptography becomes impossible. True but what we will gain from P = NP will make the whole internet look like a footnote in history.

Since all the NP-complete optimization problems become easy, everything will be much more efficient. Transportation of all forms will be scheduled optimally to move people and goods around quicker and cheaper. Manufacturers can improve their production to increase speed and create less waste. And I'm just scratching the surface.

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).

4. APPROACHES TO SHOWING P 6= NP

4.1 DIAGONALIZATION

Can we just construct an NP language L specifically designed so that every single polynomial-time algorithm fails to compute L properly on some input?

Cantor [8] showed that the real numbers are un-countable using a technique known as diagonalization. Given a countable list of reals, Cantor showed how to create a new real number not on that list.

Turing, in his seminal paper on computation [39], used a similar technique to show that the Halting problem is not computable. In the 1960's complexity theorists used diagonalization to show that given more time or memory one can solve more problems. Why not use diagonalization to separate NP from P?

Diagonalization requires simulation. Also a diagonalization proof would likely relativize, i.e., work even if all machines involved have access to the same addi-tional information.

5. CIRCUIT COMPLEXITY

To show P 6= NP it is su cient to show some NP-complete 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 [32] showed the NP-complete 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 [33]. Razborov and Rudich [34] 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.

6. PROOF COMPLEXITY

Consider the set of Tautologies, the Boolean formulas of variables over ANDs, ORs an NOTs such that every setting of the variables to True and False makes true, for example the formula

(x AND y) OR (NOT x) OR (NOT y):

A literal is a variable or its negation, i.e. x or NOT x. A formula, like the one above, is in Disjunctive Normal Form (DNF) if it is the OR of ANDs of one or more literals.

If a formula is not a tautology, we can give an easy proof of that fact by exhibiting an assignment of the variables that makes false. But if were indeed a tautology we don't expect short proofs of that. If one could prove there are no short proofs of tautology that would imply P 6= NP.

In 1985, Haken [22] showed that tautologies that encode the pigeonhole principle (n + 1 pigeons in n holes means some hole has more than one pigeon) do not have short resolution proofs.

Since then complexity theorists have shown similar weaknesses in a number of other proof systems including cutting planes, algebraic proof systems based on polynomials and restricted versions of proofs using the Frege axioms, the basic axioms one learns in an introductory logic course.

But to prove P 6= NP we would need to show that tautolo-gies cannot have short proofs in an arbitrary proof system. Even a breakthrough result showing tautologies don't have short general Frege proofs would not su ce in separating NP from P.

7. DEALING WITH HARDNESS

So you have an NP-complete problem you just have to solve. If as we believe P 6= NP you won't nd a general algorithm that will correctly and accurately solve your problem all of the time. But sometimes you need to solve the problem any-way. All hope is not lost. In this section we describe some of the tools one can use on NP-complete problems and how computational complexity theory studies these approaches. Typically one needs to combine several of these approaches when tackling NP-complete problems in the real world.

8. BRUTE FORCE

Computers have gone faster. Many moderate size problems can be solved by brute force search through all possibilities. The NP- complete travelling salesperson problem can be solved by using extensions of the cutting-plane method. 3 SAT remains NP- complete but the best algorithms can solve SAT problems on about 100 variables. For satisfiability on general formulae searching all possibilities is better. Since all these algorithms have exponential growth in their running times, even a small increase in problem size can kill an efficient algorithm. Brute Force can alone not solve NP-complete problems.

8.1 PARAMETERIZED COMPLEXITY

In a vertex cover problem, we find a set of K central people such that for every compatible pair of people, at least one of them is central. For small K we can determine whether a central set of people exists efficiently no matter the total number n of people we are considering. For the Clique problem even for small K the problem can still be diffcult.

8.2 APPROXIMATION

We can achieve a well approximate answer. In travelling salesperson problem the distances can be given as Euclidian Distance. This problem remains NP-complete but Arora [5] gives an efficient algorithm that gets very close to the best possible route. 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.

9. HEURISTICS AND AVERAGE-CASE COMPLEXITY

The study of NP-completeness focuses on how algorithms perform on the worst possible inputs. Many computer scientists employ various heuristics to solve NP-complete problems that arise from the specific problems in their fields. SAT receives more attention than any other Boolean formulas. Most natural NP-complete problems have simple efficient reductions to the satisfiability of Boolean formula. In competition these SAT solvers can often settle satisfiability of formulas of a million variables [1]. Levin [29] developed a theory of efficient algorithms over a specific distribution and formulated a distributional version of the P versus NP problem. Some problems like versions of the shortest vector problem in a lattice or computing the permanent of a matrix are hard on average exactly when they are hard on worst-case inputs, but neither of these problems is believed to be NP-complete. Whether similar worst-to-average reductions hold for NP-complete sets is an important open problem.

10. USING HARDNESS

In Section 3 we saw the nice world that arises when we assume P = NP. But we expect P 6= 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.

11. CRYPTOGRAPHY

If P = NP then public-key cryptography is impossible. Assuming P 6= NP is not enough to get public-key protocols, instead we need strong average-case assumptions about the di culty 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

12. 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 worst-case complexity assumptions. I will focus on this second approach.

13. COULD QUANTUM COMPUTERS SOLVE NP-COMPLETE 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 [20] 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.

14. FUTURE SCOPE

Ketan Mulmuley and Milind Sohoni have presented an approach to the P vs. NP problem through algebraic geometry, dubbed Geometric Complexity Theory, or GCT [30]. This approach seems to avoid the diffculties mentioned in Section 4 but requires deep mathematics that could require many years or decades to carry through.

Although all that is necessary is to show that Pn contains an integral point for all n, Mulmuley and Sohoni states that this direct approach would be difficult. Under this approach, there are three significant steps remaining:

Prove that the LP relaxation solves the integer programming problem for Pn in polynomial time, find an efficient, simple combinatorial algorithm for the integer programming problem for Pn, and prove that this simple algorithm always answers \yes."

Although step (1) is difficult, Mulmuley and Sohoni have provided definite conjectures based on reasonable mathematical analogies that would solve (1). In contrast, the path to completing steps (2) and (3) is less clear. Despite these remaining hurdles, even solving the conjectures involved in

(1) could provide some insight to the P versus NP problem.

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 6= 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 ei-ther. 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 Prof. Dhiraj Pandey 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.

[10] B. Cipra. This ising model is NP-complete. SIAM News, 33(6), July/August 2000.

[11] V. Conitzer and T. Sandholm. New complexity results about Nash equilibria. Games and Economic Behavior, 63(2):621{641, July 2008.

[12] 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.

[13] R. Downey and M. Fellows. Parameterized Complexity. Springer, 1999.

[14] J. Edmonds. Paths, trees and owers. Canadian Journal of Mathematics, 17:449{467, 1965.

[15] L. Fortnow and W. Gasarch. Computational complexity. http://weblog.fortnow.com.

[16] 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.

[17] 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.

[18] A. Razborov and S. Rudich. Natural proofs. Journal of Computer and System Sciences, 55(1):24{35, Aug. 1997.

[19] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5):1484{1509, 1997.

[20] 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.

[21] M. Sudan. Probabilistically checkable proofs. Communications of the ACM, 52(3):76{84, Mar. 2009.

[22] R. Trakhtenbrot. A survey of Russian approaches to Perebor (brute-force search) algorithms. Annals of the History of Computing, 6(4):384{400, 1984.

[23] A. Turing. On computable numbers, with an application to the Etscheidungs problem. Proceedings of the London Mathematical Society, 42:230{265, 1936.

[24] 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.