

**Submitted By:**

**PIYUSH SHARMA**

**17103110**

NP

COMPLETENESS

ABSTRACT

This report tells us about how the problem of P VS NP came into this world and how it evolved over the time. Also, it puts light on the current condition of this problem. Does P = NP? Came into this world as a great challenge of science. Also, we will see how different researchers have tried to solve this problem. We will study in detail what actually a NP and P problem is and how it is related to Turing Machine. This report also includes proofs of some famous NP Complete problems like 3 SAT problem and CLIQUE problem and how these problems were proved as NP Complete. This paper also includes the reason I chose this topic for research and my personal perception on this problem. Finally, it tells why this question of “Is P=NP?” still unsolved and what is the further scope of research in this field.

INTRODUCTION

After the computers were discovered, every programmer tried ways to find clever and easy ways so that they are able to device an algorithm which solves different problems into somewhat easy and fast way. There were some problems like Matrix Multiplication to which they found solution very easily but some problems like solving sudoku were still taking a lot of time to solve. So, they classified the problems into two types:

1. **P CLASS PROBLEMS**

* *P is a complexity class that represents the set of all decision problems that can be solved in polynomial time*.

A decision problem is one in which we are about to take a decision and the answer can be either YES or NO. For P problems, the answer is Yes or No can be decided in polynomial time.

1. **NP CLASS PROBLEMS**

* *NP is a complexity class that represents the set of all decision problems for which the instances where the answer is "yes" have proofs that can be verified in polynomial time.*

This means that if someone gives us an instance of the problem and a certificate to the answer being yes, we can check that it is correct in polynomial time.

But however, after some span of time, scientists came up with solutions to solve some problems which earlier were very hard or time-consuming to solve could be solved in polynomial time. Thus, some problems which initially were NP Class Problems shifted to P Class, which created the question into the minds of almost every scientist of “IS P=NP?”.

Then there started a debate between the intellectuals of the world about whether someday all the problems of NP Class get converted to P Class or not. This problem became so popular that it was included into the list of ‘Millennium Prize Problems’ given by The Clay Institute, United States. P = NP means that for every problem that has an efficiently verifiable solution, we can find that solution efficiently as well.

IS P=NP?

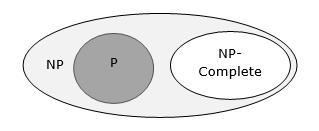
According to 80% scientists of world, P!=NP. Actually, they have seen problem which need a large time to solve. For ex: Getting to know the best move of in a game of chess. One seems to find it impossible to solve this problem. My own perception is that P!=NP. But still there exist some optimistic scientists who believe that science will take a form so smart that every problem will become easy to solve. But then problems like cryptography will also solve quickly and may cause security issues. Every problem has its own positive and negative points.

NP COMPLETE PROBLEMS

Then among the NP Class Problem, there were some problems which were extremely hard to solve and all the problems of NP Class were someway reducible to these problems. The hardest of the NP Class problems are called as NP Complete Problems.

* *NP-Complete is a complexity class which represents the set of all problems X in NP for which it is possible to reduce any other NP problem Y to X in polynomial time.*

This means that we can solve problem Y easily if we can solve X easily. Some examples of NPC problems include Partition into Triangles, Hamiltonian Cycle and 3-Coloring.



In real world, NP Completeness is a failure as nothing can be said about them. These are problems whose status is unknown. No polynomial time algorithm has yet been discovered for any NP complete problem, nor has anybody yet been able to prove that no polynomial-time algorithm exists for any of them. The interesting part is, if any one of the NP complete problems can be solved in polynomial time, then all of them can be solved.

To know whether a given problem is NP Complete or not, we need to do two things:

1. Prove that the problem is a NP Problem.
2. Prove that every problem is the NP set is polynomial time reducible to this problem.

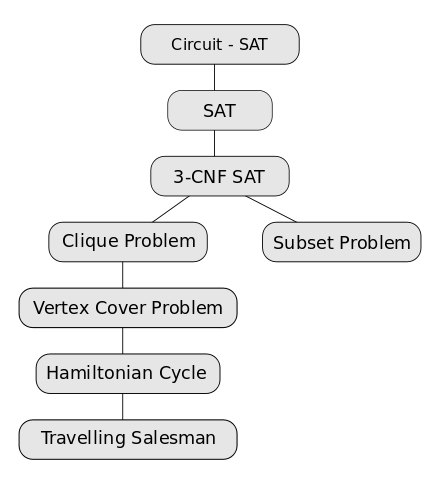
At first look it might look very difficult to reduce every problem to one single problem in polynomial time. But this problem was solved when the first NPC problem came into place. But for knowing how it was proved as NPC, we must look that what a Turing Machine is and what its function is.

There are a lot of computational programs that cannot be solved by any computer even if it is given infinite amount of time. For example: let’s talk about Turing Halting Problem (Given a program and an input, whether the program will eventually halt when run with that input, or will run forever). So, Alan Turing proved that any problem that can be solved on the deterministic Turing Machine can be solved by programming too. This statement was given way back in 1938, when even the Computers were not discovered. And this is still true that the Turing Machine is the machine with the highest computational value.

A Turing machine is a mathematical model of computation that defines an abstract machine, which manipulates symbols on a strip of tape according to a table of rules. A Turing machine can be either deterministic or non-deterministic.

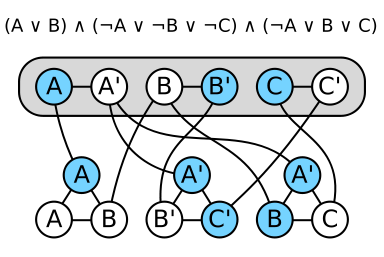
A deterministic Turing machine is the one which don’t have any choice to the action it can take for every state. Also, P problems are those which can be solved by a deterministic Turing machine in polynomial time. On the other hand, NP problems are those which can be solved by a non-deterministic machine in polynomial time.

Now, let’s prove any one problem to be NP Complete. Let’s start with the first problem i.e. 3 SAT Problem.



EVOLUTION OF NP COMPLETE PROBLEMS

3 SAT PROBLEM



In this problem, we are having some clauses made up of different literals. Then the task is to find out whether we are having any Boolean set of values for which the equation becomes true. To find this, we need to consider all the possibilities of different variables and the complexity comes out to be exponential. This problem is completely NP as if we’re having the solution, we can verify it very easily but it is very difficult to get to that decision. This was actually the first NP Complete problem defined and was used to prove many other problems to be NP Complete.

In this problem, we have a set of Boolean variables like a1, a2,…an. Now, we will define the literals like ai or NOT ai and the clause to be made up of three literals. Then we take the conjunction of clauses to know whether we have any values of the variables for which we get the output of the whole equation to be true.

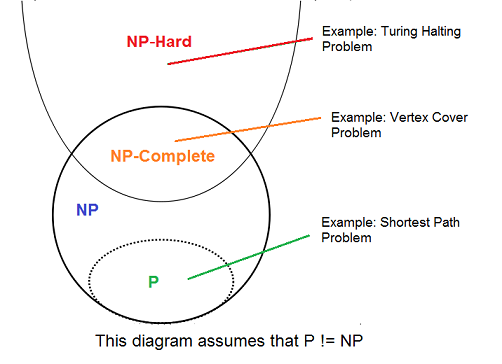
CLIQUE PROBLEM

This is a problem of finding cliques (subsets of vertices, all adjacent to each other, also called complete subgraphs) into a graph. This problem was proved to be NP Complete after the problem of Satisfiability was proved NP Complete (as given into the figure).

In this, we take every vertex to be equivalent to the different literals that are present into the 3 SAT Problem. Then their join will make the clauses and this way, the clique problem gets reduced to 3 SAT Problem. Now, as we have reduced this problem to 3 SAT problem in polynomial time, this problem also becomes NP Complete. This way all the problems are proved as NPC or NPH.

NP-HARD PROBLEM

Actually, these are issues that are the hardest problems like that of NP-complete problems. Note that NP-hard issues don't have to be compelled to be in NP, and that they don't have to be compelled to be decisions problems.



For example: Subset Sum Problem  
However, my personal favorite problem is Minesweeper problem.

ACKNOWLEGDEMENT

I would like to thank my friends for helping me in this topic and also Shilpa Mam for providing me with this topic.

REFERENCES

* <http://ijcsit.com/docs/Volume%206/vol6issue06/ijcsit2015060634.pdf>
* <https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_np_hard_complete_classes.htm>
* <https://www.geeksforgeeks.org/np-completeness-set-1/>
* <https://www.quora.com/What-is-NP-hard-and-NP-complete>
* <https://www.quora.com/How-can-I-prove-that-clique-problem-is-NP-complete>