FROM EXPONENTIAL TO POLYNOMIAL TIME COMPLEXITY

Neha Patidar
Assistant Professor(UGC NET 2017 Qualified, PhD(Pursuing))
Department of Computer Application
Faculty of Engineering and Technology
Parul University, Vadodara (Gujarat) India

Abstract: Problem statement: In this study we discuss the relationship between the known classes P and NP. We show that the difficulties in solving problem "P versus NP" have methodological in nature. An algorithm for solving any problem is sensitive to even small changes in its formulation.

Conclusion/Recommendations: As we will show in the study, these difficulties are exactly in the formulation of some problems of the class NP. We construct a class UF that contains P and that simultaneously is strictly contained in NP. Therefore, a new problem arises "P versus UF".

Key words: Class P, class NP, class UF, set-theoretic model, problem without foresight, problem that exponential in nature.

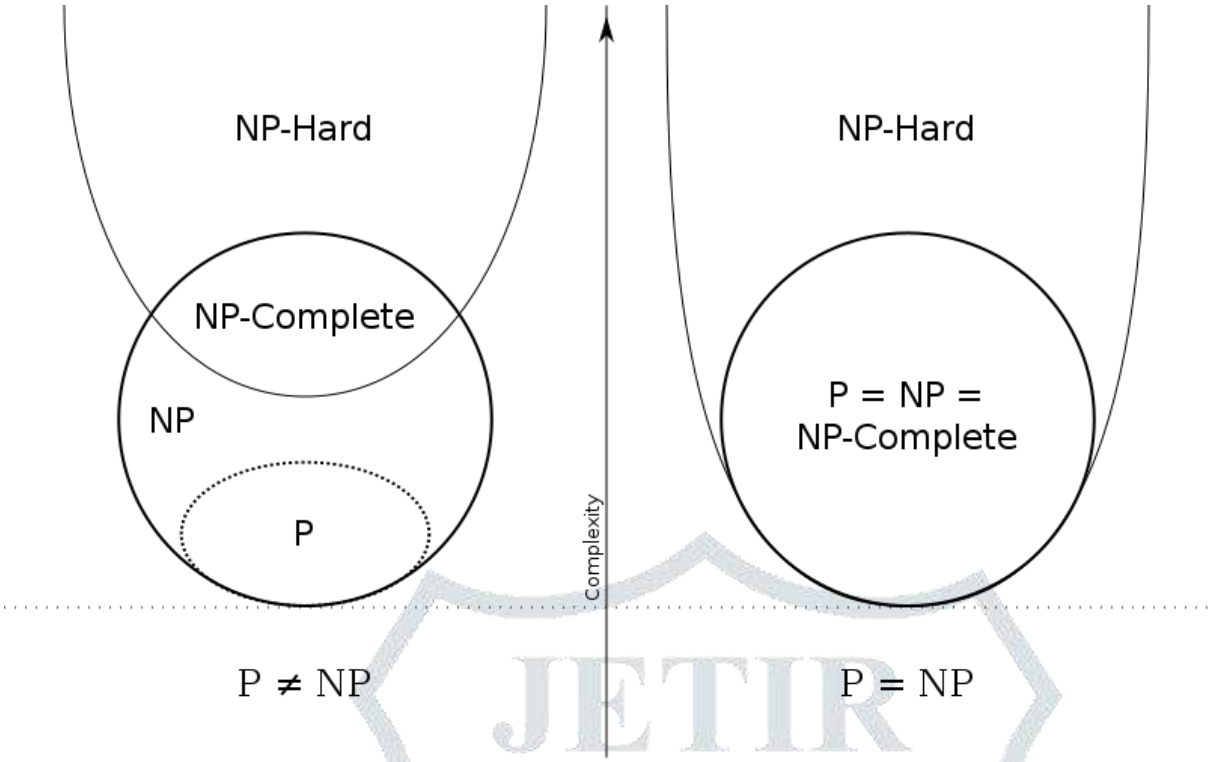
1. INTRODUCTION

The **P versus NP problem** is a major unsolved problem in computer science. It asks whether every problem whose solution can be quickly verified (technically, verified in polynomial time) can also be solved quickly (again, in polynomial time).

The informal term *quickly*, used above, means the existence of an algorithm solving the task that runs in polynomial time, such that the time to complete the task varies as a polynomial function on the size of the input to the algorithm (as opposed to, say, exponential time). The general class of questions for which some algorithm can provide an answer in polynomial time is called "class **P**" or just "**P**". For some questions, there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it is possible to verify the answer quickly. The class of questions for which an answer can be *verified* in polynomial time is called **NP**, which stands for "nondeterministic polynomial time".

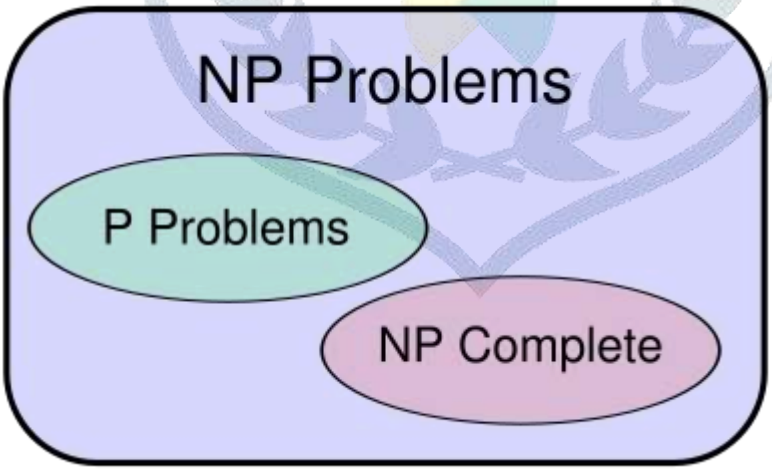
Consider Sudoku, an example of a problem that is easy to verify, but whose answer may be difficult to compute. Given a partially filled-in Sudoku grid, of any size, is there at least one legal solution? A proposed solution is easily verified, and the time to check a solution grows slowly (polynomially) as the grid gets bigger. However, all known algorithms for finding solutions take, for difficult examples, time that grows exponentially as the grid gets bigger. So Sudoku is in **NP** (quickly checkable) but does not seem to be in **P** (quickly solvable). Thousands of other problems seem similar, fast to check but slow to solve. Researchers have shown that a fast solution to any one of these problems could be used to build a quick solution to all the others, a property called **NP**-completeness. Decades of searching have not yielded a fast solution to any of these problems, so most scientists suspect that none of these problems can be solved quickly. This, however, has never been proven.

An answer to the **P** = **NP** question would determine whether problems that can be verified in polynomial time, like Sudoku, can also be solved in polynomial time. If it turned out that **P** ≠ **NP**, it would mean that there are problems in **NP** 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.



2. LADNER'S THEOREM:

In computational complexity, problems that are in the complexity class NP but are neither in the class P nor NP-complete are called **NP-intermediate**, and the class of such problems is called **NPI**. **Ladner's theorem**, shown in 1975 by Richard E. Ladner, is a result asserting that, if P ≠ NP, then NPI is not empty; that is, NP contains problems that are neither in P nor NP-complete. Since the other direction is trivial, it follows that P = NP if and only if NPI is empty.



3. MATHEMATICAL MODELS:

Let R is a finite set and Q = {R_1, R_2… R_m} is a family of nonempty subsets of R. A pair (R, Q), satisfying the property Eq. 1:

if $R_1 \in Q$ and $R_2 \subset R_1$ then $R_2 \in Q$ is called a hereditary system, or an independent system.

Let a problem Z, belonging to the class NP, is given to a hereditary system. Then, each of the subsets of $R_j \in Q$ (j =1,2,...,m) is called an admissible solution of the problem Z and every maximal respect to inclusion R_j is called support solution of the problem.

In the problem Z it is required to find a support solution that satisfies the given conditions.

The most important question that arises in the process of solving the problem, consists in determining the membership of any subset of $R_k \subset R$ to the set of admissible solutions of Q.

Consider some examples of such problems.

3.1 Satisfiability problem (SAT): Let π is a Boolean function over n variables x_1 ,...,x_n , represented in conjunctive normal form, i.e., π is a conjunction of clauses. It is required to find a satisfying assignment of variables, i.e., such that π is "true".
Let R is the set of literals r, where r is either x_i or $\overline{x}_i (i = \overline{1,n})$, belonging to some clause of the π. Then the set of admissible solutions Q contains such a subset of set R, each of which has literals of some clauses of π. To solve the question "$R_j \in Q$?" ($R_j \subset R$), it needs to determine that:
- the subset of R_j contains no alternative literal, that is, a variable x_i or $\overline{x}_i$ simultaneously
- there are clauses of π, containing together all literals from a subset R_j (that is, the subset R_j covers some of the clauses of the π)

It is clear that the SAT problem is defined on a hereditary system.

3.2 The Hamiltonian cycle problem: Let G = (V, E) is n-vertex undirected graph. It is required to find a cycle of graph edges of G, which includes each of n vertices exactly once.

We show that this problem is also defined on a Let S is the set of Hamiltonian cycles R* of G. Denote a set Q of subsets of edges of Rj (j = 1, 2, …, m) of E such that Rj $\in$ Q, if and only if Rj $\subseteq$ R*, (R* $\in$ S).Obviously, to answer on the question "Rj $\in$ Q?" it needs to determine that:
- A subset of Rj does not induce the subgraph whose vertices has degrees more than two
- There exists a Hamiltonian cycle of R* $\in$ S such that Rj $\subseteq$ R*.

Clearly, the two (E, Q) are a hereditary system.
Let (R, Q) is a hereditary system.
Let, further, w(r_i) ($i = \overline{1,n}$) is an integer, called the weight of element $r_i \in R$. For each $R_j \in Q$ we define the sum:

$$W(R_j)=\sum_{\forall r \in Rj} w(R)$$

This sum is called a weight of the subset R_j.
Suppose we want to find R* $\in$ Q, which has the maximum weight. In this case, we have formulated the optimization problem.

3.3 The Maximum Independent Set problem (MMIS):
Let G = (V, E) is a n-vertex undirected graph. It is required to find a subset V* $\subseteq$ V, which has the maximum number of vertices such that each pair of vertices in V* is not adjacent in G.

We establish that this problem is also defined on a hereditary system.
Any subset of $R_j \subseteq V$ is called independent if every two vertices of R_j are non-adjacent.
Let Q is the set of all independent sets of G. It is easy to see that the pair (V, Q) is a hereditary system. In this problem, we have the weight w(Rj) = Card(Rj) for any $R_j \in Q$. To solve the problem "$R_j \in Q$?" ($R_j \subset V$) it is sufficient to show that the vertices of R_j are pairwise non-adjacent.

It follows from the above examples that the problem of the class NP should be considered as a four
(R, Q, P, f), where R is the set of all elements of the solution, Q is the set of admissible solutions of the problem, P is the system of predicates which determine that a Rj $\subset$ R is an admissible solution, i.e., Rj $\in$ Q and f is the map: Rj $\rightarrow$ W that defines the "weight" of the admissible solution. In the future, we believe that if (Rj) can be computed in polynomial time from value Card (Rj).
Thus, we have defined a set-theoretic model of a problem of NP. In computational complexity theory, each problem of NP is considered as problem recognition. A recognition problem is a computational problem, whose solution is "yes" or "no" (Tucker, 1997). The solution of computational problem (in this case, a support solution) can be considered as "proof" that the corresponding recognition problem of NP has an answer "yes". Therefore, the concept of an admissible solution is broader than the concept of "proof" for the recognition problem.

3.4 Sequential method: Let there be a problem Z $\in$ NP. We assume that Z = (R, Q, P, f) is determined on a hereditary system (R, Q). The question arises: How can we construct an admissible solution Z? A Turing machine is a general model of computation (see, for example, (Aho *et al*., 1974; Tucker, 1997). Therefore, we can assume that we have a single-tape Turing machine M. The machine M processes symbols in the cells of the tape sequentially, i.e., symbol by symbol. If we assume that the Turing machine solves the problem Z then we can consider the result of each step of M as an admissible solution Z. It is natural to consider recording a symbol on the tape as the constructing the next element of the admissible solution.

Thus, the procedure constructing any admissible solution Rj $\in$ Q (j = 1, 2… m) is extensive at the time, i.e., its elements are obtained sequentially, element by element.

The method of constructing the required solution, when we obtain it elements step by step, element by element, we call sequencer (Plotnikov, 1997; 2011).
Let R1, R2 $\in$ Q is admissible solutions of the problem Z $\in$ NP such that R1 $\subset$ R2. We denote the time of their constructing as t (R1) and t (R2) respectively.

Theorem 1: t (R1) < t (R2).

Proof: Without loss of generality, we can assume that Card (R2) = Card (R1) + 1. By definition of the sequential method, the admissible solution R2 can be obtained after the construction of a Turing machine of the admissible solution R1. This will require at least one cycle of the machine. This proves Theorem 1.

Theorem 2: The solution of any problem Z ∈ NP can be obtained by a sequential method.
Proof: We believe that every problem of the class NP is solvable, i.e., each such problem can be solved by a deterministic Turing machine. Since this machine runs sequentially, it produces the solution step by step, an element of the element. Therefore, Theorem 2 is true.
Obviously, we can assume that the sequential method is solely the general method for solving each problem Z ∈ NP.

Indeed, for example, suppose we need to find some independent set graph vertices. Any independent set of graph vertices is an admissible solution of MMIS.
However, only the maximal independent set is the support solution of MMIS. The global solution of this problem-the maximum independent set-is also some support solution.

Obviously, in the common case, the simultaneous choice of several independent vertices is not possible if the structure of the graph is not known in advance. It is clear that every next subsequence of vertices can be selected only if we know what vertices have already been selected in an independent set earlier.

Problems without foresight: Let there be a problem Z∈ NP, determined by the hereditary system (R, Q). Let, further, R1 ∈ Q is an admissible solution that is not supported. In accordance with forming the next admissible solution R2 such that

R1 ⊂ R2 and Card (R2) = Card (R1) + 1

The problem of NP can be divided into two classes (Plotnikov, 1997; 1999):

- The problems for which the next solution R2 = R1∪ {r} (r ∈ R) can be found in the polynomial time by means of joining to the R1 of one element of the set R\R1
- All the other problems of NP In other words, the problem of NP can be classified according to the computation time of the predicate "R1 ∪ {r} ∈ Q?" for any admissible solution R1∈ Q and for each element r ∈ R\R1. If such predicate can be computed in the polynomial time from the dimension of the problem then this problem is called a problem without foresight. Otherwise, the problem is called exponential in nature.

Note that the selected term-problem that is exponential in nature-should not be taken literally. As is usual in the complexity theory (see, for example, (Garey and Johnson, 1979) such problems may require O (n!) or O (n^n) steps to compute the predicate "R1 ∪ {r} ∈ Q?". The set of all problems without foresight will be denoted by UF, where UF ⊆ NP.

Theorem 3: The support solution Z ∈ NP can be found in the polynomial time if and only if the Z ∈ UF.
Proof: Let there be a problem Z ∈ NP such that Z ∈ UF. By the definition of problems without foresight, each next admissible solution of Z can be found in the polynomial time. Since ∅ ∈ Q for any problem Z ∈ UF and the support solution contains no more than p (n) of elements, where n is the dimension of the problem and p (n) is a polynomial, this implies the polynomial time of building the support solution.

On the other hand, let there be the problem Z ∈ NP is such which solvable in the polynomial time. Assume that Z ∉ UF. In this case, there exists at least one admissible solution of Z, found at the exponential time. By condition of Theorem 3, the support solution of problem Z ∈ UF is found in the polynomial time. We have a contradiction of Theorems 1 and 2.

Thus, the class UF induced problems of NP for which the support solution can be constructed in the polynomial time. In common cases, each support solution is not the desired solution of the problem.

Theorem 4: UF ⊂ NP and UF ≠ NP.
Proof: By definition, UF ⊆ NP. Definition of problems in the class NP does not exclude such problems, for which the verification (obtaining) some intermediate results may require exponential time. In problems of the class UF all the intermediate results can be obtained only in polynomial time. Therefore, UF ≠ NP. Taking into account Theorems 3 and 4, we have the following result.

Corollary 1: P ⊂ UF and P ≠ NP. Using our terminology, we can reformulate the third item in the definition of the class NP as follows.
Time t for verifying the obtained support solution is some polynomial function from
n : t= φ(n).
Then we have following definition of the class UF. A problem Z belongs to a class UF If:

- The problem can be defined by a finite number n of symbols
- The problem solution can be represented by a finite number m of symbols, where m is a polynomial function of n: m = f (n)
- The time t for verifying any admissible and finite solution is some polynomial functions of n: t =φ(n)

As we can see the definition of the class UF differs from the definition of the class NP only the third item. The definition of the class NP wider than the problems of the class UF since the class NP can includes and objectives for which the time for verifying any admissible solutions is exponential, i.e., requires a lot of exhaustive search. Thus, the class NP includes all problems of the class UF. It is easy to see that the class of P is entirely in the class UF.

Thus, a direct answer to the problem of "P vs NP" consists in the fact that the class P is not equal to the class NP.

4. CONCLUSION

When the question of the relationship between classes P and NP was formulated, many researchers believed that this will solve the problem with the possibility to develop (or not) an effective (polynomial time) algorithm for all problems of the class NP.

However, we found that the existing definition of class NP is redundant, i.e., It allows including in this class of problems that are exponential in nature. As we earlier removed the problems with exponential length solutions from the class NP, now, we remove from the class of NP the problems that are exponential in nature.

In other words, it is expedient to focus on the study of problems without foresight (the class UF) for searching an effective solution algorithm.

However, in this case, we have two questions. The first question concerns the problems which are exponential in nature. We see that such practical important problems as the Hamiltonian cycle problem (in the current formulation) cannot be solved effectively. However, it is necessary take into account that the set of problems without the foresight includes such NP-complete problems as the satisfiability problem and the maximum independent set problem.

However, every problem of the class NP can be reduced to some (NP-complete) problem without foresign in the polynomial time, for example, to the satisfiability problem (Garey and Johnson, 1979). Such reduction can be considered as a reformulation of the relevant problem that is exponential in nature into the problem without the foresight.

The second question consists that we must be reformulated the old problem of "P vs NP" into a new one, namely: "P vs UF".

Thus, unfortunately, the elucidation of the relationship between classes P and NP does not answer to the question about the possibility of effectively solving problems as the class NP and the class UF. We just found that the lack of an existing definition of the class NP consists that we do not determine the polynomial time by checking the truth of the predicate P for the problem $(R, Q, P, f) \in NP$.

To have a hope for constructing effective solution algorithm, we need to consider the problem of the class UF. Obviously, that the determining properties of these problems is a possibility to construct a support solution in the polynomial time. If the problems Z1 is out the class UF then we must reduce it into a Z2 from UF, the global solution of which is also a global solution Z1. Sometimes we cannot say that a problem belongs to UF or not. In this case, we have little knowledge about this problem to determine its existence in the class UF. Then it is expedient to suspect that the problem lies outside of the class UF.

5. REFERENCES:

- Aho, A.V., J.E. Hopcroft and J.D. Ullman, 1974. The Design and Analysis of Computer Algorithms. 1st Edn., Addison-Wesley Publishing, New York, pp: 470.
- Erusalimsky, J.M., 2000. Discrete Mathematics .1st Edn., Vuzovskaya Kniga, Moscow, pp: 280.
- Garey, M.R. and D.S. Johnson, 1979. Computers and Intractability: A Guide to the Theory of NPCompleteness. 1st Edn., W.H. Freeman, San Francisco, ISBN-10: 0716710447, pp: 338.
- Tucker, A.B., 1997. The Computer Science and Engineering Handbook. 1st Edn., CRC-Press, Inc., Boca Raton, Florida, ISBN-10: 0849329094, pp:2656.