Description on the P vs. NP
Provided by Andrew Nassif
A Guide to its Mathematics

ABSTRACT

Most people when thinking of the P vs. NP thinks of it as unsolved or unsolvable although it goes with a simple algorithmic proof in which I have written in my previous paper. Now as you may know, the P vs. NP can lead to some of the greatest breakthroughs in computer science, and mathematical physics. The question however asks you to use a method to figure out the complexity vs. non-complexity of a problem, which I have demonstrated in my last paper as the example of someone removing 100 possible students from a group of 400 and then coming up with the different combinations in which the 300 can't be picked. This problem can be used in example with probability and solved through the complex equation where there is a solution for P equaling NP and you would then get ((400!)-(100!*3)). I gave this example using set theory and logic as well. [1]This number would then be more than the number of atoms in the known planet in which we live on. However this is just one of the examples.My next example was an example of how some computers show through algorithms and polynomials that P does equal NP.

THE BASICS

Now let us take a step backwards and go to the basics. What are Polynomials? What are Algorithms? How are they used in coding? Polynomials expression are the size and input of Algorithms, this is how they are related. Algorithms are for example simple codings, they tell the computer basic step by step solutions into solving problems, kind of like html but more advanced and in relation to Python and mathematical coding. Algorithms are known to be effective in measuring a finite list and calculation the function or output and inputs of a simple problem. They can be used in a computer or simply by hand in order to measure the complexity of a problem by solving it.

Now let us think to ourselves what is time complexity? How are these methods used in solving the P vs. NP? Time complexity quantifies the amount of time taken by an algorithm to run as a function to the length of taken up time in order to solve that particular function. These methods can be used to measure the complexity of a problem throughout how much time is taken. The main problem in the P vs. NP is to measure the complexity vs. non complexity of a problem through a complete proof. Now that we figured out this part, how do we disprove Ladner's Theorem.

DISPROVING LADNER'S THEOREM

Ladner's Theorem is also known as the NP-Intermediate. The problem states that neither P or NP is complete in terms of polynomials and natural algorithms. [2]The problem with this Theorem is that what Ladner didn't realize is that for every complex problem there could possibly be a less complex way to solve it through circular logic and measuring out your non-possibilities.

---

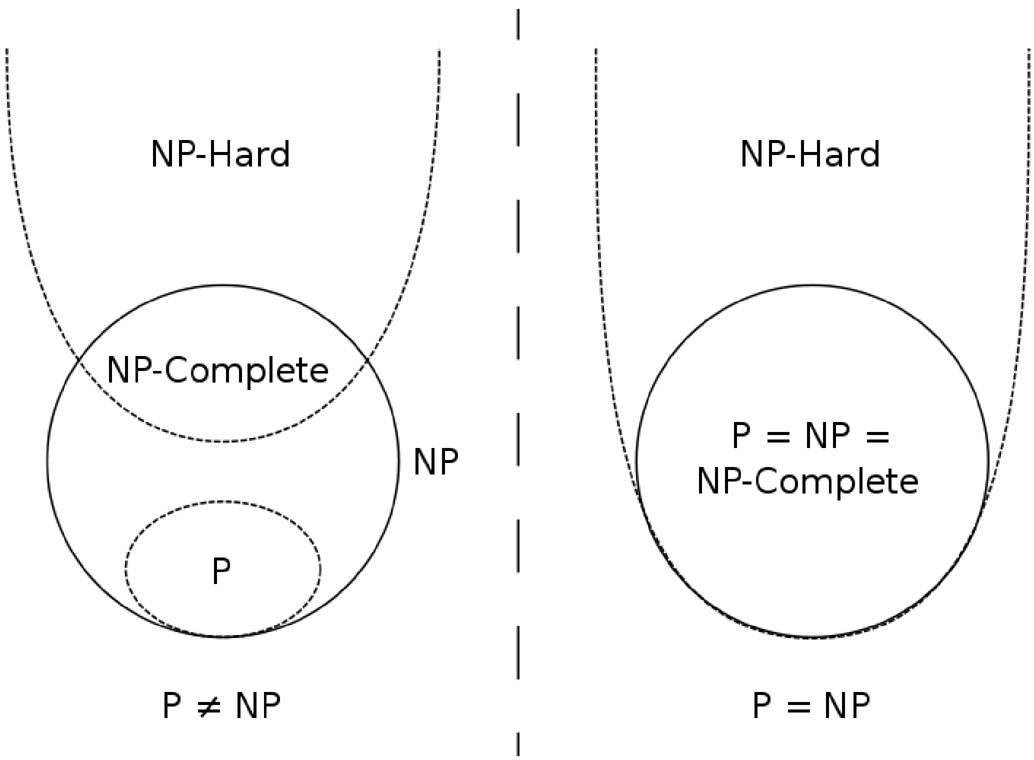
[1] **^** Lance Fortnow. Computational Complexity Blog:Complexity Class of the Week: Factoring. 13 September 2002.

[2]**^** Geere, Duncan. "'Travelling Salesman' movie considers the repercussions if P equals NP". Wired. Retrieved 26 April 2012.

^ Berger B, Leighton T (1998). "Protein folding in the hydrophobic-hydrophilic (HP) model is **NP**-complete". *J. Comput. Biol.* **5** (1): 27–40. doi:10.1089/cmb.1998.5.27.PMID 9541869.

Description on the P vs. NP
Provided by Andrew Nassif
A Guide to its Mathematics

Ladner's considers P and NP both NP-Intermediate. Trying to solve a problem through NPI, he considers the problem as artificial in terms of mathematics, as well as a discrete logarithm. Now here comes Ladner's incorrectness. The main problem is that he tried disproving it using incorrect measurements of the P vs. NP, rather than the measurements of the P vs. NP through normal use of algorithms and polynomials.



^ Berger B, Leighton T (1998). "Protein folding in the hydrophobic-hydrophilic (HP) model is **NP**-complete". *J. Comput. Biol.* **5** (1): 27–40. doi:10.1089/cmb.1998.5.27.PMID 9541869.