# NP and P Packaged in Complexity

Byron Oswaldo Ullauri

In this note, we explore an important concept within the field of mathematics and computer science that is ubiquitous, namely the NP problems. We begin with algorithms that are constantly used to solve various problems; an example being those that require complicated data processing. In such cases, mathematicians and computer scientists have not always been able to solve or find the most effective way of approaching a problem. As a result, a significant part of both fields requires the study of problems as related to the algorithms by which they are solved and tested. By definition, an algorithm describes a method that can efficiently solve a problem. In computational terms, this efficiency refers to the amount of time it takes to run the algorithm relative to the input used, formally known as an algorithm's time complexity. In relation to this, it's generally agreed upon that problems which are solvable within polynomial time, a run time time resulting from "a simple polynomial function of the size of the input" (6), use efficient algorithms to do so. The overlying theory that accompanies these statements is known as Complexity Theory. It states that their level of difficulty can classify problems. The different possible classes that problems can fit into include NP, P, NP-Hard, and NP-Complete problems.

NP-Problems, or Nondeterministic Polynomial, describe the class of problems that can be solved and verified as correct within polynomial time. In other words, this class encompasses problems that can be solved through the use of inefficient methods and proof, but can be efficiently verified as having a valid proof. An example of such a problem is found in a case where you are given an array of numbers and are asked if it is possible to split it into two parts that when added form an equal sum (3). In order to solve this problem, you would have to check numerous subsets, then add the numbers on each side, and see if both sums are equal. Although you will eventually reach a conclusion, depending on the size of the array, the amount of time it would take to iterate through this would be increasingly inefficient. On the other hand, if you were able to split the array into two equal sums, checking your approach and solution would just involve a couple steps of adding and comparing the sum on each side, thus making it possible to efficiently verify it.

P-Problems are a subset of problems within NP such that they encompass problems that can be both solved and verified efficiently. The P in this class stands for polynomial time referring to the number of steps used in the algorithm. As previously mentioned, algorithms which are solved within polynomial time are considered efficient. Consequently, P-Problems are also considered NP since their solution is already efficient, meaning the solution's verification would not require

the need for a proof. Examples of P-Problems include those found in basic arithmetic or finding the digits in the value of Pi.

NP-Hard problems are defined as being those whose solution can be slightly modified into solving any NP-Problem. This means that a problem is considered NP-Hard when there exists an NP problem that can be reduced to it (within polynomial time) (7). According to Wolfram MathWorld, an NP-Hard problem is described as being "at least as hard as any NP-problem." Examples of NP-Hard problems also include the subset sum problem mentioned before and a problem known as the "Traveling Salesman Problem," which will be defined as NP-Complete.

Finally, the last class of problems is known as NP-Complete. They comprise problems which are both NP and NP-Hard. As a result, these types of problems are what are considered the most "difficult" NP problems. An example of an NP-Complete problem is the "Traveling Salesman Problem." It states that a salesman passing through n cities must find a path resulting in the least total distance traveled if he/she wants to be the most efficient. No general method of solution has been found although proposed solutions have been made within them, solutions make use of the Hamiltonian Cycle, "a closed loop through a graph that visits each node exactly once" (1). Instead, the only verifiable way of solving this problem is to check all possible solutions, which at best results in exponential time (4).

Ultimately, Complexity Theory gives rise to a question that is still unanswered: are all NP-problems actually P-problems? The underlying principle behind this question is that if it is true, then all problems we consider NP actually have efficient algorithms that we have yet to discover. This phenomenon is known as the P vs NP question. In the past, a Linear Programming problem thought to be NP was proven to be P by Leonid Khachiyan. His ground-breaking ellipsoid algorithm found ways to minimize convex functions using iterative methods which when applied to linear optimization was found to be highly effective (5). The fact that Khachiyan was able to change a problem's NP classification leaves the P vs NP question completely open for debate.

References


1. http://mathworld.wolfram.com/
2. https://en.wikipedia.org/wiki/Time_complexity
3. https://cs.stackexchange.com/questions/9556/what-is-the-definition-of-p-np-np-complete-and-np-hard/9566#9566
4. https://www.mathsisfun.com/sets/np-complete.html
5. https://en.wikipedia.org/wiki/Ellipsoid_method
6. Google definition
7. https://en.wikipedia.org/wiki?curid=54681

Nominating Faculty: Professor Satyanand Singh, Mathematics 2540, Department of Mathematics, School of Arts and Sciences, New York City College of Technology, CUNY.

Cite as: Ullauri, B.O. (2017). NP and P packaged in complexity. *City Tech Writer, 12,* 85-86. Online at https://openlab.citytech.cuny.edu/city-tech-writer-sampler/