CSCI3390-Lecture 18: Why is the $\mathbf{P} =?\mathbf{NP}$ Problem Such a Big Deal?

The conjecture that $\mathbf{P}$ is different from **NP** made its way on to several lists of the most important unsolved problems in *Mathematics* (never mind *Computer Science,* where it is *the* most important unsolved problem). It is relatively simple to state, even simpler to paraphrase:

Is finding a solution harder than just verifying a solution?

and, judging by the strenuous efforts that have made to solve it during the past four decades, very, very hard. Depending on whom you listen to, the conjecture is either something that *must* be true, with unthinkable consequences otherwise—or, something that might very well be false.

If it is false, then it will reshape our view of the computational universe, in a way that goes beyond the development of efficient algorithms for a collection of **NP**-complete combinatorial decision problems like CLIQUE and 3-COLORABILITY. These notes are a summary of a few of these consequences.

1 Polynomial solution to optimization problems

If **P=NP** then we would have polynomial-time solutions for a number of hard optimization problems, not just decision problems. We'll concentrate on one such problem, but the principle is quite general.

The *Traveling Salesman Problem* (TSP) has as its input a *weighted* undirected graph G, together with a home vertex v. The weights are integers attached to each edge of the graph. Thus the size of a problem instance includes the total number of bits required to write down all the weights, as well as tabulate all the vertices and edges. The output of the problem is a circuit, beginning and ending at vertex v, that has minimum total weight. Think of the vertices of the graph as

the cities on a salesman's route, and the weight of an edge between, say, Boston and Philadelphia as the cost of traveling between these two cities. The problem is then to visit every city on the salesman's route as efficiently as possible.

First note that this problem is as hard as any problem in **NP**, in the sense that if we had a polynomial-time solution, then it would follow that **P=NP**. To see this, let G be a graph with n vertices, and let us assign weight 1 to every edge. Pick any vertex v as the home. Then this weighted graph has an optimal traveling salesman circuit of length n if and only if G has a Hamiltonian circuit. Thus a polynomial-time solution to the TSP implies a polynomial-time solution to an **NP**-complete decision problem, and thus **P=NP**.

We'll show the converse: That is, suppose that **P=NP**. We will give a polynomial-time algorithm that finds an optimal route for the salesman. Let N be an integer that is greater than the number of vertices, the number of edges, and the number of bits in each weight. We can take N as a measure of the size of the input instance. The graph has no circuit at all if it is not connected, and this is something that we can determine in polynomial time. If the graph is connected, then doing a depth-first search would provide a circuit that traverses each edge exactly twice. The total weight of this circuit is no more than twice the sum of N N-bit numbers; that is, no more than $M = 2N \cdot 2^N$.

Now consider the following decision problem (Problem 1): The input consists of the TSP input, together with a threshold value T. The output is 'Yes' if and only if there is a circuit of weight no more than T. This problem is easily seen to be in **NP**, since if we had a candidate circuit we could efficiently verify that its weight does not exceed the threshold. Thus if **P=NP** we have a polynomial-time algorithm for this decision problem. We now couple this algorithm with binary search, first testing if there is a circuit with weight no more than $M/2$. If there is, we repeat with $T = M/4$, if not, with $T = 3M/4$. In no more than $log_2 M < 2N$ iterations, we will find the exact weight of the optimal path. Since each iteration requires time $p(N)$, where p is a polynomial (using our **P=NP** assumption!) we have shown a polynomial-time algorithm for finding the exact weight of an optimal circuit.

What about finding the optimal circuit itself? We now turn to another decision problem (Problem 2). Here the input is the weighted graph G, a home vertex v, a sequence of no more than N vertices $v = v_0, v_1, \ldots, v_k$, and a target T. The output is 'Yes' if and only if there is a circuit of weight T that *starts with* the sequence $v_0, \ldots, v_k$. Once again, this problem is in **NP**, so if **P=NP**, it can be decided in polynomial time.

We now use for T the optimal value determined by our solution to Problem 1,

and try out every vertex of G for v_1. We will thus run our algorithm no more than N times to determine the first vertex on an optimal circuit. We proceed similarly to find an appropriate value of v_2, etc. Thus we find an optimal circuit in N^2 iterations of our algorithm for Problem 2, so this is still polynomial time.

2 coNP, $\forall\exists\mathbf{P}$, and beyond ('collapse of the polynomial-time hierarchy')

Consider the problem of determining whether a propositional formula ϕ is a *tautology*–that is, whether every assignment is satisfying. This is equivalent to the problem of determining whether $\neg\phi$ is a *contradiction–i.e.,* has no satisfying assignments. The naïve algorithm for testing this is, as with SAT, to try out every one of the 2^n assignments to the n variables in the formula. However, with SAT there was the possibility of getting lucky and chancing upon a satisfying assignment before all the possibilities are examined. With the tautology problem, or the equivalent contradiction problem, there are no such lucky guesses to show that something is a tautology, so 'yes' instances of this problem seem harder than SAT. (On the other hand, we could get lucky and find a non-satisfying assignment which would give us a quick 'No' answer.)

The contradiction problem is the *complement* of SAT. Similarly, the complement of the graph 3-colorability problem is the collection of graphs that *can't* be 3-colored; the complement of the Hamiltonian circuit problem is the collection of graphs that *don't* have Hamiltonian circuits, *etc.* The class **coNP** consists of the complements of **NP** problems. The tautology problem is also in here, because it is the complement of the **NP** problem of determining if there is a non-satisfying assignment.

If it happened that $\mathbf{P} = \mathbf{NP}$, then $\mathbf{NP}$ would be closed under complement (because **P** is). We would then have $\mathbf{NP} = \mathbf{coNP} = \mathbf{P}$, so all of these complement problems would also be solvable in polynomial time.

But there's more. Suppose we are given two propositional formulas (or even boolean circuits) ϕ and ψ and asked if they are equivalent–i.e., whether they agree on all assignments. That is, we are asking whether

$$(\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$$

is a tautology. As noted, this is a problem is in **coNP**. To solve it, we would presumably test every possible assignment.

Now, suppose we are given a single formula or circuit ϕ, and we ask if there is a *smaller* formula or circuit ψ that is equivalent to ϕ. Ordinarily we would have to run through every formula smaller than ϕ and test it for equivalence, which requires us to run through every assignment. But now we can apply the hypothesis **P=NP** twice: equivalence testing would be in **P**, so the existence of a witness formula ψ would be in **NP**, and hence in **P**. So this problem, too, which appears to be at a higher level of complexity, would also be solvable in **P**.

*(More technical).*What are these higher levels of complexity? We can define these various compleA problem L in **NP** have the following form: There is some polynomial-time computable, boolean-valued function F and positive$k > 0$ such that

$$w \in L \Leftrightarrow \exists x(|x| \leq |w|^k \wedge F(w, x)).$$

This is just a fancy way in the language of predicate logic to express the existence of a polynomial-size witness. We might paraphrase this and write

$$\mathbf{NP} = \exists \mathbf{P}.$$

Then **coNP** is characterized by

$$w \in L \Leftrightarrow \forall x(|x| \leq |w|^k \rightarrow F(w, x)),$$

or, more succinctly,

$$\mathbf{coNP} = \forall \mathbf{P}.$$

Continuing with this theme, our formula equivalence problem has another layer of quantifiers:

$$\exists \psi(|\psi| \leq |\phi| \wedge \forall \mathbf{x}(\mathbf{x} \in \{0,1\}^n \rightarrow \psi(\mathbf{x}) = \phi(\mathbf{x}))),$$

where by $\phi(\mathbf{x})$ I mean the result of substituting the assignment **x** for the variables in ϕ. We might call the underlying complexity class

$$\exists \forall \mathbf{P}.$$

The same argument we gave above shows that, if **P=NP**, then no matter how many levels of quantification we apply, all such problems are solvable in polynomial time. This is called the *collapse of the polynomial-time hierarchy.*

3 The End of Creativity in Mathematics and Science?

This point was raised, in different language, by Gödel in a letter written in the 1950's—long before the research on **NP**-complete problems began—and only discovered after his death.

A sentence of arithmetic expressing a complex theorem, if written out completely, might take up perhaps one thousand (10^3) symbols. Any proof that we are likely to comprehend, or, for that matter, check with a computer, would not use more than one billion symbols (10^9). Now consider the set of sentences ϕ of arithmetic that are theorems, and have proofs of length no more than $|\phi|^3$. This problem is in **NP**. If **P=NP** then we would have a polynomial-time algorithm for settling all such questions of arithmetic, and even finding the proofs. Some would view this as a complete automation of the process of discovery and creation in mathematics—and this has even been extrapolated to the discovery of scientific theories. (I don't find this terribly convincing, and I don't really buy it, so I may not be the best advocate or explainer of this line of thought.)