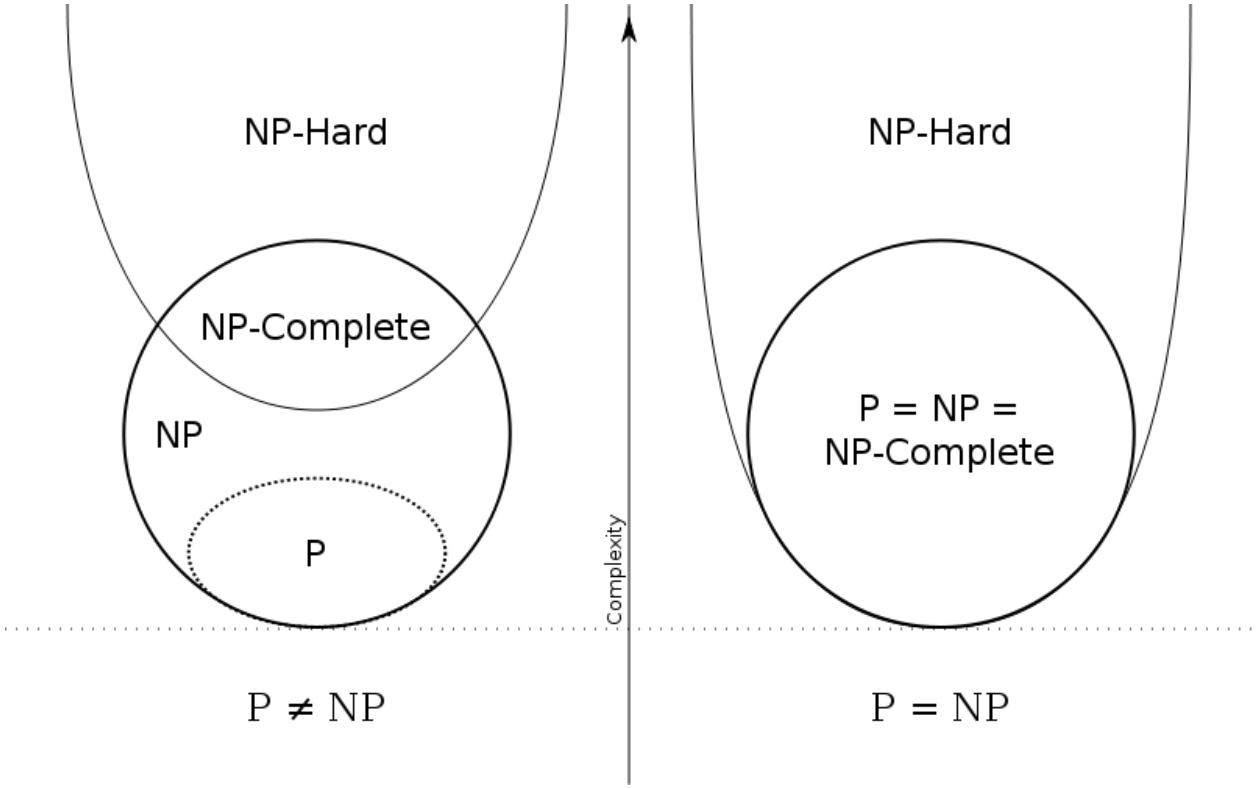
P versus NP: Is the world as we perceive it?

Imagine that you are working for your favorite company when, suddenly, an amazing idea for a product hits you. The company you work for agrees to give you a set amount of time to visit each of your prospective customers who live in different cities, so you need to find out if there is a route that allows you to travel to all of those customers and then back to where you work in the set amount of time. A simple solution is to just search through every possible route and check if one is under the time limit. Let's count the number of possible orderings of cities given you have just three prospective customers. You have to choose which of the three cities you will go to first. You then have to choose which of the two remaining cities you will go to next. You then know which city is third because it is the only one remaining, and you know that you will be going home from there. Therefore, the number of possible routes is the product of three, two, and one: six. You should surely be able to check all of these quickly. If you have four customers the number of possible routes is the product of four, three, two, and one: 24. And it follows that if you have 5 customers, there are 120 possible routes; there are 720 possible routes with only 6 customers. Clearly the number of possible routes is the factorial function, which grows extremely quickly. With just ten customers, there are over 3 and a half million routes to check, but this can still be solved with a computer. However, as the number of customers grows larger, the problem quickly becomes impossible for a computer to solve in our lifetime. This is an example of an NP-complete problem known as a version of the traveling salesman problem. NP-complete problems are very difficult to solve, but it's very easy to verify if a proposed solution is correct or not. A very important problem in computer science known as the P versus NP problem essentially asks if these NP-complete problems are as difficult as they seem. Hardesty's, Fortnow's, Gasarch's, Ferguson's, and Root-Bernstein's works help explain why the important P versus NP problem has remained unsolved for a long time.

Hardesty helps explain what the P versus NP problem is. He writes that "P is a set of relatively easy problems, and NP is a set of what seem to be very, very hard problems, so P = NP would imply that the apparently hard problems actually have relatively easy solutions" [1]. This helps give a good intuition for the problem, but the actual definition is a bit more complicated. NP is actually the set of all problems whose answers can be verified in polynomial time. Polynomial time simply means that the amount of time the algorithm takes is proportional to a polynomial function of the size of the algorithm's input. A polynomial time algorithm is much faster than a non-polynomial time algorithm. In other words, for a problem to be in NP, the problem must be quickly checkable; that is a solution can quickly be determined correct or incorrect. For example, in the traveling salesman problem presented earlier, if I gave a proposed solution (a route), you'd be able to quickly tell me if that route is under the time limit or not. Therefore, that problem is in the class NP.

Contrastingly, there is no known way of actually solving the traveling salesman problem quickly, so it isn't believed to be in P. This is because P is the set of all problems whose solutions can be found in polynomial time. In other words, for a problem to be in P, the problem must be quickly solvable. That is all problems in NP are quickly checkable, and all problems in P are quickly solvable. Here's an example

of a problem in P: given a list of numbers, find the minimum number. Clearly this problem can be solved by simply looking at every number once and keeping track of the minimum number; the amount of operations this takes is directly proportional to the amount of numbers in the list. However, this problem is also quickly checkable; if I were to give you a number you'd be able to check if it were the minimum or not by simply comparing it to all the numbers in the list. This means that this problem is also in NP. In fact, every problem in P is in NP, so P is a subset of NP. This makes sense because it's at least as hard to come up with a correct solution as it is to verify a solution. The "very, very hard problems" Hardesty was referring to are actually NP-complete problems. These are problems that exist in NP but are too difficult to be in P as far as we know. The P versus NP problem is essentially asking if P equals NP (P and NP are the exact same set) or if P doesn't equal NP (P is just a proper subset of NP). The set representation of these two scenarios can be seen below [2].



Fortnow, Hardesty, and Gasarch all mention that solving the P versus NP problem would have widespread implications. Similar to the traveling salesman problem presented earlier, there are many optimization problems in real life that have to do with scheduling that are NP-complete. If P equals NP, then all of these optimization problems would become much more efficient [3]. Hardesty mentions the traveling salesman problem and the "fairly common [...] scheduling tasks" that are NP-complete [1]. All of these would be revolutionized if P equals NP. There is one frightening result that all three of the authors mention; if P equals NP, public-key encryption would no longer work [1], [3], [4]. This effectively means that data on the web would no longer be secure; the whole field of cryptography would have to be reinvented. If P were equal to NP, the world be fundamentally different.

Gasarch's 2012 poll [4] confirms that the majority opinion in computer science is that P does not equal NP. 83% of responders said that they believe P is not equal to NP while only 9% said that P equals NP, and 0.6% said they don't know. In contrast, in Gasarch's 2002 poll [5] on the same topic, 61% said that P is not equal to NP, 9% said that P equals NP, and 22% said they don't know. It appears that as time has gone on, more people have gravitated toward the belief that P is not NP. When only looking at the "bigshots" of the field (those who have won either a Turing Award, Fields Medal, Nevanlinna Prize, Godel Prize, Kanellakis award, Knuth Prize, or are in the National Academy of Science or Engineering), 81% said that P is not equal to NP, 9% said that P = NP, and 9% said they don't know. Therefore, the opinion of the most renowned computer scientists is consistent with the opinion of the whole group. In his poll, Gasarch also asked when people think the P versus NP problem will be solved. Because most believe that P doesn't equal to NP, this question is similar to asking when it will be proven that P doesn't equal NP. In 2002, 62% of responders thought that the problem would be resolved by 2100 and only 17% thought it would be resolved later than 2100. Contrastingly, the numbers in the 2012 poll are 53% and 41% respectively. It seems that people are becoming more pessimistic regarding how long it will take to solve the problem. Gasarch also asked people what techniques they think will be used to solve the P versus NP problem. By far the most popular answer was "New Techniques and Hard Math." This suggests that solving the problem will require some innovative thinking.

Root-Bernstein and Ferguson both strongly emphasize the importance of creativity in furthering knowledge in technical fields. Ferguson views the mind's eye as paramount in innovation in technical fields [6] while Root-Bernstein analyses the correlation between many great scientists and their inclination toward creative work [7]. Creativity is very relevant in the P versus NP problem because the problem has remained unsolved for so long despite the overwhelming opinion that P should not be the same as NP. Because solving the P versus NP problem will most likely require new techniques, it is clear that those attempting to solve it will need be quite innovative and really think creatively. If the P versus NP problem were going to be solved by traditional thinking, it would have be solved by now.

Although the P versus NP problem is unsolved and may very well remain unsolved for a long time, there is hope for NP-complete problems in approximation algorithms. The version of the traveling salesman problem that we were looking at can be solved to within a few percentage points of the correct solution in nearly polynomial time [8]. Similarly, there are polynomial-time approximation algorithms for special cases of difficult scheduling problems that yield solutions that are no more than 1.5 times the best solution [9]. Although P is most likely not equal to NP (so we probably won't ever have algorithms that find the best solution to general case NP-complete problems in polynomial time), these approximation algorithms serve as a fantastic consolation.

Reference List

1. L. Hardesty. “Explained: P vs. NP.” Internet: http://newsoffice.mit.edu/2009/explainer-pnp, Oct. 29, 2009 [Nov. 23, 2014].
2. B. Esfahbod. “Euler diagram for P, NP, NP-Complete, and NP-Hard set of problems.” Internet: http://upload.wikimedia.org/wikipedia/commons/a/a0/P_np_np-complete_np-hard.svg, Nov. 1, 2007 [Dec. 1, 2014].
3. L. Fortnow. “The Status of the P Versus NP Problem.” Internet: http://cacm.acm.org/magazines/2009/9/38904-the-status-of-the-p-versus-np-problem/fulltext, Sep. 2009 [Nov. 23, 2014].
4. W. I. Gasarch “The Second P =? NP Poll.” Internet: https://www.cs.umd.edu/~gasarch/papers/poll2012.pdf, 2012 [Nov. 23, 2014].
5. W I. Gasarch “The P =? NP Poll.” Internet: http://www.cs.umd.edu/~gasarch/papers/poll.pdf, 2002 [Nov. 23, 2014].
6. E. S. Ferguson, Engineering and the Mind’s Eye. Cambridge, Massachusetts: The MIT Press, 1992
7. R. S. Root-Bernstein, Visual Thinking: The Art of Imagining Reality. Philadelphia, Pennsylvania: American Philosophical Society, 1985
8. A. Gupta et al. “Approximation Algorithms for Optimal Decision Trees and Adaptive TSP Problems.” Internet: http://arxiv.org/pdf/1003.0722v2.pdf, Mar. 2010 [Dec. 1, 2014].
9. K Jansen et al. “Faster Approximation Algorithms for Scheduling with Fixed Jobs.” Internet: http://theory.epfl.ch/osven/Ola%20Svensson_publications/cats10.pdf, Jan. 2011 [Dec. 1, 2014].