Complexity Upper and Lower Bounds

William Schultz

February 9, 2022

For any decision problem, we can establish both *upper bounds* and *lower bounds* on its complexity. Recall that a decision problem is formulated in terms of a language L, consisting of a set of strings. The decision problem for a given language L is to determine whether $w \in L$ for some given string w.

An upper bound makes a statement about the maximum hardness/complexity of the problem, and a lower bound makes a statement about the minimum easiness of the problem. Establishing an upper bound is typically much easier, since you only need to provide a concrete algorithm that solves the problem in some worst case running time (i.e. show there exists an algorithm). Establishing a lower bound is generally much harder, since you need to show that there exists no algorithm that can solve the problem more efficiently than a certain complexity class (i.e. show no algorithm exists).

For example, one of the best known algorithms for 3-SAT as of 2019 has a numerical upper bound of something around $O(1.307^n)$ [**?**]. There exists no known, general algorithm that can solve 3-SAT in polynomial time. But, it has also not been proven that such an algorithm doesn't exist. It seems that the best known lower bounds for SAT sit somewhere in the polynomial range of $n^{1.801}$, though this has some other caveats about "time-space tradeoffs" which I don't fully understand [**?**]. Proving that SAT, for example, had an exponential (or even super polynomial) lower bound would, of course, establish that $P \neq NP$, since SAT is NP-complete, and this would serve to separate P from NP. Of course, one could also prove $P = NP$ by simply giving a polynomial time algorithm for SAT i.e. by dropping the upper bound from exponential to polynomial. This might be "easier", in the sense that you would only have to find a single algorithm, but it may be "harder" in the sense that P=NP may not actually be true!