NP Hardness

Algorithms & Theory

Nathanael Seen

June 15, 2021

Introduction

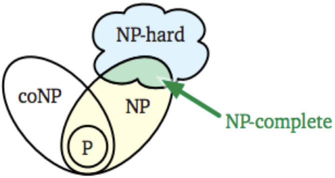
- So far, all problems we studied in this module had polynomial-time algorithms, for example
  - Searching problem: Linear search $O(n)$, Binary search $O(\log n)$
  - Sorting problem: Merge sort $O(n \log n)$, Bubble sort $O(n^2)$
  - Single-Source Shortest Path problem: Dijkstra $O(V \log E)$, Bellman-Ford $O(VE)$
- However, there are problems which are 'very hard' for computers to solve
- In fact, some problems are non-computable, like the Halting problem
- In this series of slides, we shall only look at computable problems, and in particular focus on the concept of NP-hardness

Problem

- Consider the longest-path problem on a positively-weighted graph $G = (V, E)$, where we want to find a simple path (with no cycles) from start node s to destination node t, of at least $k > 0$ edges, which incurs maximal cost
- Clearly, the shortest path algorithms won't be of much help
- Even though we negate all edges, and try to run Dijkstra's algorithm, this could work but only if the original graph had negative edges
- Try finding a polynomial-time algorithm!

Explanation

- Formally, a problem is in the 'NP'-class if a solution to that problem (given its inputs) can be verified in polynomial-time
- Clearly, all problems we have studied in CS2040 are NP-complete, because we can find a solution to those problems in polynomial-time by simply running algorithms learnt so far to solve them!
- In fact, since there exists polynomial-time algorithms to solve these problems, they are more accurately in the 'P'-class:

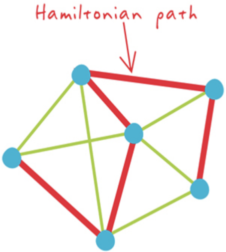


Explanation (cont.)

- Problems in ‘NP’ don’t really have a polynomial-time algorithm to solve them, in a sense that no one has been able to find any yet!
- There are a few famous problems which have already been proven to be NP-hard; 3-SAT, Graph-coloring, Hamiltonian path
- To show that a problem α is NP-hard, we need to show that there exists an efficient (polynomial-time) *reduction* from any of the known problems (in NP) to α
- Hence, the longest-path problem we just seen can be shown to be NP-hard, by showing that there exists a polynomial-reduction from the Hamiltonian path problem

Explanation (cont.)

- To show that longest-path is NP-hard
  - We need to find a polynomial-reduction; ϕ, from the original input graph instance G to Hamiltonian path, such that G has a Hamiltonian path if and only if $G' = \phi(G)$ has a longest path of at least length $k' = \phi(G)$
  - A Hamiltonian path is one that visits exactly all nodes in the graph exactly once (hence there would be $|V| - 1$ edges in total)



Explanation (cont.)

- The reduction; ϕ, is obvious, set $G' = \phi(G) = G$ and $k' = \phi(G) = |V| - 1$, and it is clearly polynomial incurring $O(V)$ to iterate through all vertices to compute $|V|$
- ($\Longrightarrow$) Now, suppose G has a Hamiltonian path, then this path starting at s ending at t would have length $|V| - 1$. Since, $G' = G$, this path is also on G'. Since all weights are positive, this path has to be the longest-path in G', because any path of length lesser than $|V| - 1$ would have total weight lesser than the current path. Thus G' has a longest-path of length at least $k' = |V| - 1$. ■
- ($\Longleftarrow$) Conversely, suppose G' has longest-path of length at least k', then the path has to be of length exactly $k' = |V| - 1$, else we would not get a simple path (with no cycles). Since, $G = G'$, this path is also on G. But, since this path is of length $k' = |V| - 1$, it is a Hamiltonian path. ■
- **Hence, longest-path problem is NP-hard!**

References

- https://en.wikipedia.org/wiki/Longest_path_problem
- https://www.quora.com/If-we-negate-the-edge-weights-in-a-graph-G-V-E-and-then-run-Bellman-Ford-does-this-compute-longest-paths
- https://www.csie.ntu.edu.tw/~lyuu/complexity/2016/20161129s.pdf
- https://baniel.github.io/algorithm/DanielAlgorithm7.1/
- CS3230 Lecture 9 and 10 notes (Prof. Divesh)