

# Intractability and Undecidability

September 22, 2019

## Decision Problems

- **Decision Problem:** Given a problem instance  $I$ , answer a certain question in "yes" or "no"
- **Problem Instance:** Input for a specific problem.
- **Problem Solution:** Correct answer ("yes" or "no") for the specified problem instance.  $I$  is a **yes-instance** if the correct answer for the instance  $I$  is "yes".  $I$  is a **no-instance** if the correct answer for the instance  $I$  is "no"
- **Size of a problem instance:**  $\text{Size}(I)$  is the number of bits required to specify (or encode) the instance  $I$ .

## The Complexity Class $P$

- **Algorithm Solving a Decision Problem:** An algorithm  $A$  is said to solve a decision problem  $\Pi$  provided that  $A$  finds the correct answer ("yes" or "no") for every instance  $I$  of  $\Pi$  in finite time.
- **Polynomial-time Algorithm:** An algorithm  $A$  for a decision problem  $\Pi$  is said to be a polynomial-time algorithm provided that the complexity of  $A$  is  $O(n^k)$ , where  $k$  is a positive integer and  $n = \text{Size}(I)$ .
- **The Complexity Class  $P$**  denotes the set of all decision problems that have polynomial-time algorithm solving them. We write  $\Pi \in P$  if the decision problem  $\Pi$  is in the complexity class  $P$ .
- For example, normally most of questions we are confronting with are in  $P$ , like finding the maximum/minimum number in an array, sorting a given array, etc

**The Complexity Class  $NP$**  Before jumping to NP problem, we need to learn a new concept – Certificate

- **Certificate:** Informally, a certificate for a yes-instance  $I$  is some "extra" information  $C$  which makes it easy to **verify** that  $I$  is a yes-instance.
  - **Certificate Verification Algorithm:** Suppose that **Ver** is an algorithm that verifies certificates for yes-instances. Then **Ver**( $I, C$ ) outputs "yes" if  $I$  is a yes-instance and  $C$  is a valid certificate for  $I$ . If **Ver**( $I, C$ ) outputs "no", then either  $I$  is a no-instance, or  $I$  is a yes-instance and  $C$  is an invalid certificate.
  - **for every** yes-instance  $I$ , **there exist** a certificate  $C$  such that **Ver**( $I, C$ ) outputs "yes".
  - **for every** no-instance  $I$ , **for every** certificates  $C$ , **Ver**( $I, C$ ) outputs "no".
- **The Complexity Class  $NP$**  denotes that the set of all decision problems that have polynomial-time certificate verification algorithms solving them. We write  $\Pi \in NP$  if the decision problem  $\Pi$  is in the complexity class NP.
- Note that we are NOT required to find the certificate in polynomial time, we just need to make sure that verification algorithm runs in polynomial time.
- $P \subseteq NP$

### The Complexity Class $NP-C$

- It denotes that the set of all decision problems  $\Pi$  that satisfy the following two properties:
 
$$\Pi \in NP$$

$$\text{For all } \Pi' \in NP, \Pi' \leq_p \Pi$$
- NP-C is an abbreviation for NP-complete.
- $A \leq_p B$  indicates that problem B is harder than A and we say A can be reduced to B.
- **Note that** the definition does not imply that the existence of NP-complete problem!
- Travelling salesman problem(TSP) is a classical NP-complete problem.
- Example for proof of reduction:

Prove that the following problem is NP-complete. Given two graphs,  $H = (V_H, E_H)$ , and  $G = (V_G, E_G)$ , is  $H$  a subgraph of  $G$ , i.e. is there a mapping  $\pi$  of the vertices of  $H$  to the vertices of  $G$  such that  $\pi$  is one-to-one (it never maps two vertices of  $H$  to the same vertex of  $G$ ) and such that for every pair of vertices  $u, v \in V_H$ , we have  $(u, v) \in E_H$  iff  $(\pi(u), \pi(v)) \in E_G$ .

Firstly, we need to show that SUBGRAPH is in NP.

Certificate: providing a set of vertices and edges in graph  $G$ .

Verification algorithm:

We trace through the entire subgraph  $H$  to check whether it is contained in graph  $G$ .

Run time: Trance throught the entire graph takes  $\Theta(V + E)$  which is polynomial.

Therefore SUBGRAPH is in NP.

Now we need to show that SUBGRAPH is in NPC.

Consider we use *Clique* as our reduction.

$$CLIQUE \leq_p SUBGRAPH$$

We will define the transformation  $f$  as follows:

$f$  takes an integer  $i$  and a graph  $G$ .

It outputs a graph the original  $G$  and paired with  $k_i$  subgraph.

We will show that the transformation of  $f$  is polynomial.

Adding edges and vertices to a graph takes constant time.

Therefore the transformation can be in polynomial time, thus  $f$  is a polynomial transformation.

Now we need to show that the reduction is correct.

Let  $(i, G)$  be a yes-instance to clique.

Then we have  $f(i, G) = (k_i, G)$ .

Since  $(i, G)$  is a yes-instance to clique.

Therefore we know that  $G$  contains a complete subgraph with degree  $i$ .

Therefore, we know that  $G$  contains  $k_i$  graph.

Thus  $(k_i, G)$  is a yes-instance to subgraph.

Let  $(H, G) \in Image(f)$  be a yes-instance to subgraph problem.

Then by construction of  $f$ , we know that  $H$  is a  $k_i$  graph.

Since  $(k_i, G)$  is a yes-instance to SUBGRAPH.

Therefore we know that  $G$  contains a  $k_i$  graph.

Thus  $G$  contains a complete subgraph of degree  $i$ .

Then  $(i, G)$  is a yes-instance to CLIQUE.

Since CLIQUE is NPC, and CLIQUE can be reduced to SUBGRAPH therefore SUBGRAPH is also a NPC problem.

## NP-hard Problem

- A problem  $\Pi$  is NP-hard if there exist a problem  $\Pi' \in NPC$  such that  $\Pi' \leq_p^T \Pi$
- Every NP-complete problem is NP-hard, but there exist NP-hard problems that are not NP-complete.
- Typically examples of NP-hard problems are optimization problems corresponding to NP-complete decision problems.

A image is good for understanding the relationship between above complexity classes:

