NP-Completeness

Sadoon Azizi

s.azizi@uok.ac.ir

Department of Computer Engineering and IT

Spring 2019

Classification of the Problems

- **Unsolvable:** There is no algorithm to solve them (Halting Problem)
- **Intractable:** As they grow large, we are unable to solve them in reasonable time (Hamiltonian Cycle)
- **Tractable:** We are able to solve them in reasonable time (Sorting)

Q: What constitutes reasonable time?

A: Standard working definition: ***polynomial time***

  - On an input of size n, the worst-case running time is $O(n^k)$ for some constant k
  - Polynomial time: $O(n^2)$, $O(n^3)$, $O(1)$, $O(n \lg n)$
  - Not in polynomial time: $O(2^n)$, $O(n^n)$, $O(n!)$

P vs. NP

- **P:** The class of problems, for which a Polynomial-time algorithm exists.
  - **Ex:** fractional knapsack, finding maximum subarray, rod cutting, etc.
- **NP:** The class of problems for which a solution can be ***verified*** in polynomial time
  - **Ex:** Hamiltonian cycle, graph coloring, 3SAT, etc.

Decision vs. Optimization

- **Decision problem:** a decision problem is a problem that can be posed as a yes-no question of the input values.
- **Optimization problem:** optimization problems are concerned with finding the best answer to a particular input.

Decision vs. Optimization

Example (Knapsack Problem)

Suppose that we have n objects, say o_i ($i = 1, 2, \cdots, n$), each with corresponding weight (w_i) and profit (p_i), and a weight bound b.

- Optimization: Find an $X = (x_1, x_2, \cdots, x_n)$ that maximize $\sum_{i=1}^{n} x_i p_i$ with respect to $\sum_{i=1}^{n} x_i w_i \leq b$.
- Decision: For a given k, is there a feasible solution, say $X = (x_1, x_2, \cdots, x_n)$, where $\sum_{i=1}^{n} x_i p_i \geq k$?

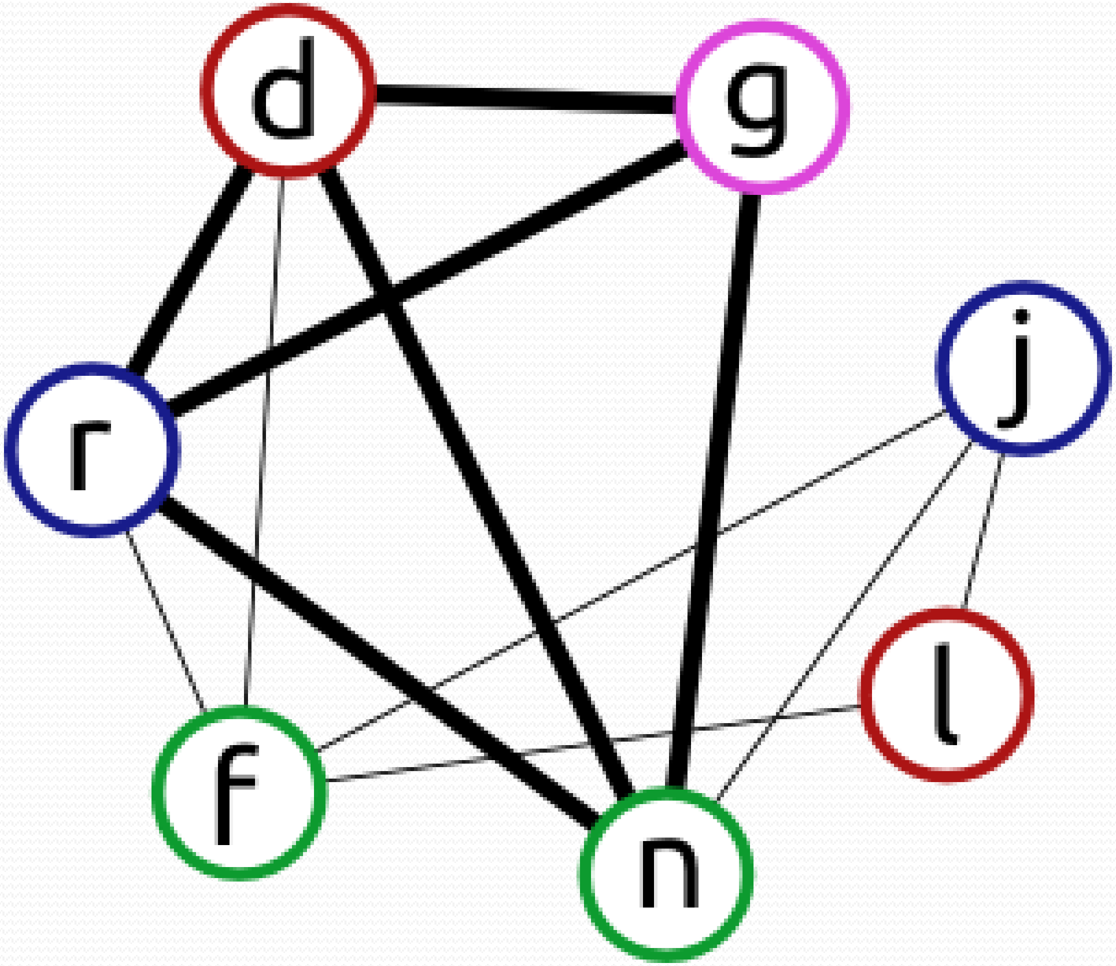
Decision vs. Optimization

Example (Max-Clique Problem)

Suppose that a graph $G = (V, E)$ is given.

- Optimization: Find a maximal subset $V' \subseteq V$, such that the induced graph by V' is complete graph (The clique is a complete subgraph).
- Decision: For a given k, is there a subset $V' \subseteq V$ with $|V'| \geq k$, such that the induced graph by V' is complete graph (a clique of size at least k)?

Decision vs. Optimization



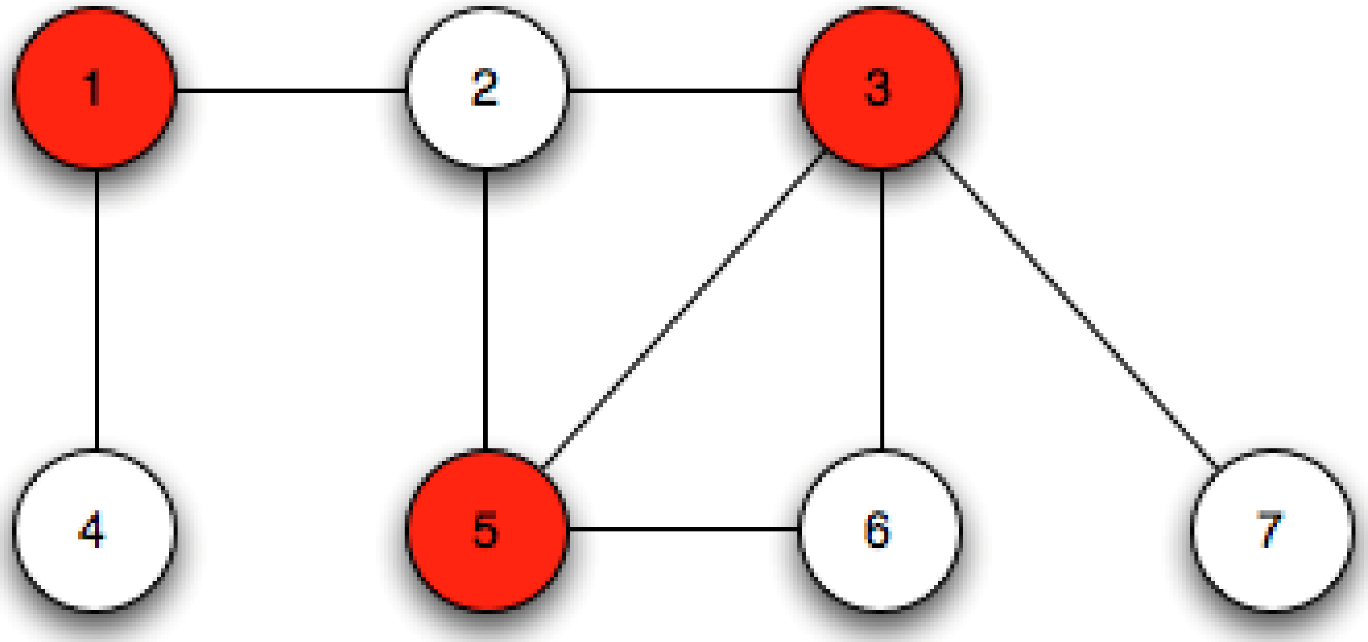
Decision vs. Optimization

Example (Min-Vertex Cover Problem)

Suppose that a graph $G = (V, E)$ is given.

- Optimization: Find a minimal subset $V' \subseteq V$, such that for each $e = (v_1, v_2) \in E$, either $v_1 \in V'$ or $v_2 \in V'$ (V' covers the E).
- Decision: For a given k, is there a subset $V' \subseteq V$ with $|V'| \leq k$, such that V' covers the E?

Decision vs. Optimization

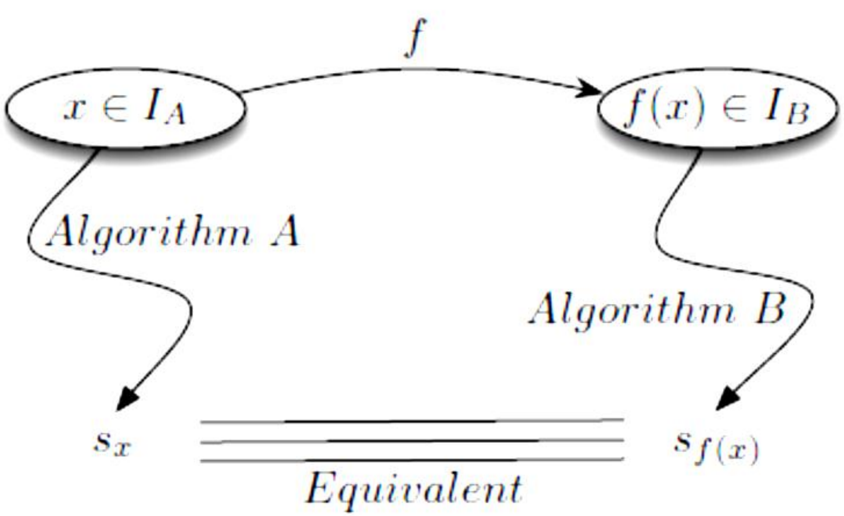


The principle of Reduction

Definition

Suppose that A and B are two decision problems. We say that A is reduced to B (denoted by $A \preccurlyeq_P B$), if there exists a polynomial algorithm, say f, such that:

- $x \in Instance(A) \Longrightarrow f(x) \in Instance(B)$.
- x is a yes-instance of $A \Longleftrightarrow f(x)$ is a yes-instance of B.



Application of reduction

The reduction defines an order over the decision problems with respect to their level of difficulties.

- If B is easy to solve, then A is also easy.
- Conversely, If A is hard to solve, then B is also hard.

The theory of NP-Completness

Definition

A problem L is called $NP-$Hard if all NP problems are reduced to L, i.e.

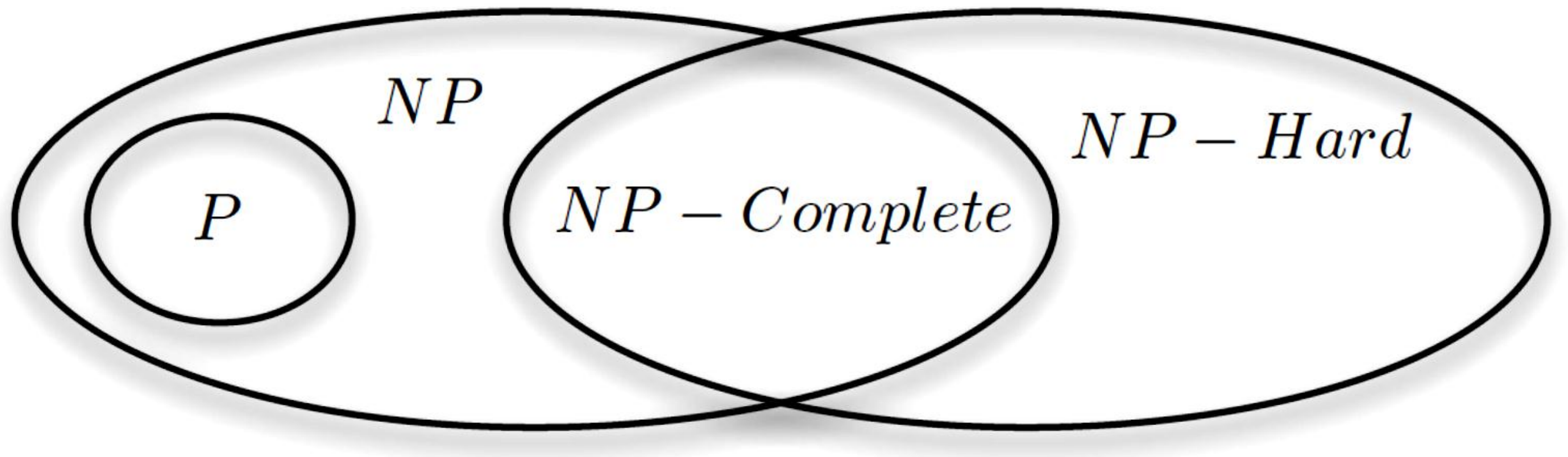
$$L \in NP - Hard \iff \forall L' \in NP \ : \ L' \preccurlyeq_P L.$$

Definition

A problem L is called $NP-$Complete if all NP problems are reduced to L and $L \in NP$, i.e.

$$L \in NP - Complete \iff L \in NP \cap NP - Hard.$$

The theory of NP-Completness



The theory of NP-Completeness

- NP-Complete problems are the "hardest" problems in NP:
  - If any *one* NP-Complete problem can be solved in polynomial time…
  - …then *every* NP-Complete problem can be solved in polynomial time…
  - …and in fact *every* problem in **NP** can be solved in polynomial time (which would show **P = NP**)
  - Thus: solve Hamiltonian-cycle in O(n^{100}) time, you've proved that **P = NP**. Retire rich & famous.

Travelling Salesman Problem (TSP)

- The **travelling salesman problem (TSP)** asks the following question:

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

- **A very important question:**

- Which complexity class does the TSP belong to? (NP-complete or NP-hard?)