NP-Completeness (part 3 of 3)

Outline

1. Introduction
2. P and NP
3. NP-complete (NPC): formal definition
4. How to prove a problem is NPC
5. How to solve a NPC problem: approximate algorithms

IV. How to prove a problem is NP-complete

- The reducibility relation is *transitive*.

- To prove that a problem $A \in$ NP is NPC, it suffices to prove that some other NPC problem B is *polynomially reducible* to A:

  Step 1: choose some known NPC problem B
  Step 2: define a polynomial transformation T from B to A
  Step 3: show that $B \leq_T A$

- Why? the logic is as follows:

  Since B is NPC, all problems in NP is reducible to B.
  Show B is reducible to A.
  Then all problems in NP is reducible to A.
  Therefore, A is NPC

IV. How to prove a problem is NP-complete

Examples:

1. Directed HC $\leq_T$ Undirected HC (Next)
2. Subset-Sum $\leq_T$ Job Scheduling (Handout)
3. Graph 3-COLOR $\leq_T$ 4-COLOR (Homework #8)
4. Subset Sum $\leq_T$ Set Partition (Homework #8)

$$\left\{ \begin{array}{l} \text{Directed HC} \\ \text{Subset-Sum} \\ \text{Graph 3-COLOR} \end{array} \right\} \text{ are NPC} \Longrightarrow \left\{ \begin{array}{l} \text{Undirected HC} \\ \text{Job Scheduling} \\ \text{4-COLOR} \\ \text{Set Partition} \end{array} \right\} \text{ are NPC.}$$

IV. How to prove a problem is NP-complete

Example:

- The directed HC is known to be NPC.
- Show that

$$\text{directed HC} \quad \leq_T \quad \text{undirected HC}$$

- Therefore we conclude that the undirected HC is also NPC.

IV. How to prove a problem is NP-complete

Example, cont'd:

- Define transformation:

  Let $G = (V, E)$ be a directed graph. Define G to the undirected graph $G' = (V', E')$ by the following transformation T:

  - $\underline{v \in V} \quad \longrightarrow \quad \underline{v^1, v^2, v^3 \in V' \text{ and } (v^1, v^2), (v^2, v^3) \in E'}$
  - $\underline{(u, v) \in E} \quad \longrightarrow \quad \underline{(u^3, v^1) \in E'}$

- T is polynomial-time computable.

- Show that

$$G \text{ has a HC} \Longleftrightarrow G' \text{ has a HC.}$$

IV. How to prove a problem is NP-complete

Example, cont'd:

"$\Rightarrow$" Suppose that G has a directed HC: $v_1, v_2, \ldots, v_n, v_1$
Then

$$v_1^1, v_1^2, v_1^3, v_2^1, v_2^2, v_2^3, \ldots, v_n^1, v_n^2, v_n^3, v_1^1$$

is an undirected HC for G'.

"$\Leftarrow$"

1. Suppose that G' has an undirected HC, the three vertices v^1, v^2, v^3 that correspond to one vertex from G must be traversed **consecutively** in the order v^1, v^2, v^3 or v^3, v^2, v^1, since v^2 **cannot** be reached from any other vertex in G'.
2. Since the other edges in G' connect vertices with superscripts 1 or 3, if for any one triple the order of the superscripts is 1, 2, 3, then the order is 1, 2, 3 for all triples. Otherwise, it is 3, 2, 1 for all triples.
3. Therefore, we may assume that the undirected HC of G' is

$$\underline{v_{i_1}^1, v_{i_1}^2, v_{i_1}^3}, \underline{v_{i_2}^1, v_{i_2}^2, v_{i_2}^3}, \ldots, \underline{v_{i_n}^1, v_{i_n}^2, v_{i_n}^3}, \underline{v_{i_1}^1}.$$

Then

$$v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_{i_1} \text{ is a directed HC for } G.$$

IV. How to prove a NP-complete problem

Sketch for Problem 5 of Homework #8: show that

$$\text{Subset-Sum} \leq_T \text{Set-Partition}$$

- Let S be an instance of Subset-Sum with $w = \sum_{s \in S} s$ and the target c.
- Define the set S' (i.e., the transformation T from S to S') as follows:

  $$S' = S \cup \{u, v\},$$

  where

  $$u = 2w - c, \quad v = w + c.$$

- next to show that

  S is a Yes-instance of Subset-Sum $\Longleftrightarrow$ S' is a Yes-instance of Set-Partition

---

Subset sum decision problem: Given a positive integer c, and the set $S = \{s_1, s_2, \ldots, s_n\}$ of positive integers s_i for $i = 1, 2, \ldots, n$. Assume that $\sum_{i=1}^{n} s_i \geq c$. Is there a $J \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in J} s_i = c$

IV. How to prove a NP-complete problem

Hint for Problem 5 of Homework #8, cont'd':

$\Longrightarrow$ Let $J \subseteq S$ and the elements in J sum to c. Then $J \cup \{u\}$ sum to $2w$. Note that the elements in $\overline{J} = S - J$ sum to $w - c$. Hence, $\overline{J} \cup \{v\}$ also sums to $2w$. Therefore, S' can be partioned into $J \cup \{u\}$ and $\overline{J} \cup \{v\}$ where both partitions sum to $2w$. We conclude that a Yes-instance of Subset-Sum transforms to a Yes-instance of Set-Partition.

$\Longleftarrow$ Assume S' can be partitioned into two sets, A and $\overline{A} = S' - A$, such that

$$\sum_{x \in A} x = \sum_{x \in \overline{A}} x. \tag{1}$$

Since $w + u + v = 4w$, the sum of the elements in both sets must be equal to $2w$. Therefore, u must be in one set and v must be in the other because $u + v = 3w$. Without loss of generality, let $u \in A$. Then

$$2w = \sum_{x \in A} x = u + \sum_{x \in A - u} x = 2w - c + \sum_{x \in A - u} x.$$

It implies that

$$\sum_{x \in A - u} x = c$$

Thus, we conclude that a Yes-instance of Set-Partition transforms to a Yes-instance of Subset-Sum.

V. How to solve a NPC problem

Example 1: Bin Packing problem

Suppose we have an unlimited number of bins, each of capacity 1, and n objects with sizes $s_1, s_2, \ldots, s_n$, where $0 < s_i \leq 1$.

- Optimization problem: Determine the smallest number of bins into which objects can be packed and find an optimal packing.
- Decision problem: Do the objects fit in k bins?

Theorem. Bin Packing problem is NPC (reduced from the subset sum).

V: How to solve a NP-complete problem

Approximate algorithm for the Bin Packing

- *First-fit strategy (greedy):*

  places an object in the first bin into which it fits.

- Example: Objects $= \{0.8, 0.5, 0.4, 0.4, 0.3, 0.2, 0.2, 0.2\}$
- *First-fit strategy* solution:

B_1	B_2	B_3	B_4
		0.2	
0.2	0.4	0.3	
0.8	0.5	0.4	0.2

- Optimal packing:

B_1	B_2	B_3
	0.2	0.2
0.2	0.3	0.4
0.8	0.5	0.4

V. How to solve a NP-complete problem

Theorem. Let $S = \sum_{i=1}^{n} s_i$.

1. The optimal number of bins required is at least $\lceil S \rceil$
2. The number of bins used by the first-fit strategy is never more than $\lceil 2S \rceil$.

V. How to solve a NP-complete problem

The vertex-cover problem:

- A vertex-cover of an undirected graph $G = (V, E)$ is a subset set of $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ (inclusive) or $v \in V'$.
- In other words, each vertex "covers" its incident edges, and a vertex cover for G is a set of vertices that covers all edges in E.
- The size of a vertex cover is the number of vertices in it.
- Decision problem: determine whether a graph has a vertex cover of a given size k
- Optimization problem: find a vertex cover of minimum size.
- **Theorem.** The vertex-cover problem is NPC.

V. How to solve a NP-complete problem

The vertex-cover problem:

- An approximate algorithm
  $C = \emptyset$
  $E' = E$
  while $E' \neq \emptyset$
  let (u, v) be an arbitrary edge of E'
  $C = C \cup \{u, v\}$
  remove from E' every edge incident on either u or v.
  endwhile
  return C
- **Theorem.** The size of the vertex-cover is no more than twice the size of an optimal vertex cover.

VI–V recap

1. How to prove a problem is NP-complete
   4 case studies
2. Approximate algorithms for solving NPC problems:
   2 case studies