Lecture 24: NP-Completeness Proofs

(CLRS 34.5.1)

June 24th, 2002

1 NP-Completeness

- We have been discussing *complexity theory*
  - classification of problems according to their difficulty
- We introduced the classes P, NP and EXP

EXP	$=$	{Decision problems solvable in exponential time}
P	$=$	{Decision problems solvable in polynomial time}
NP	$=$	{Decision problems where YES solution can verified in polynomial time}

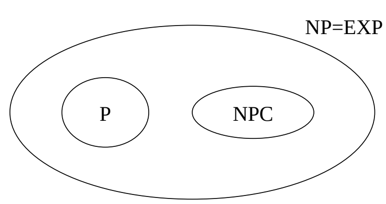
- A major open question in theoretical computer science is if $P = NP$ or not.
- We also introduced the notion of *polynomial time reductions* $X \leq_P Y$:

> A problem X is polynomial time reducible to a problem Y ($X \leq_P Y$) if we can solve X in a polynomial number of calls to an algorithm for Y (and the instance of problem Y we solve can be computed in polynomial time from the instance of problem X).

- We then introduced the class of NP*-complete* problems NPC

> A problem Y is in NPC if
> a) $Y \in NP$
> b) $X \leq_P Y$ for all $X \in NP$

  and discussed how the problems in NPC are the hardest problems in NP and the key to resolving the $P = NP$ question.
  - If one problem $Y \in NPC$ is in P then $P = NP$.
  - If one problem $Y \in NP$ is not in P then $NPC \cap P = \emptyset$.
  - By now a lot of problems have been proved NP-complete
  - We think the world looks like this—but we really do not know:



- If someone found a polynomial time solution to a problem in NPC our world would "collapse" *and* a lot of smart people have tried really hard to solve NPC problems efficiently

  $\Downarrow$

  We regard $Y \in NPC$ a strong evidence for Y being hard!

2 NP-Complete Problems

- The following lemma helps us to prove a problem NP-complete using another NP-complete problem.

Lemma: If $Y \in NP$ and $X \leq_P Y$ for some $X \in NPC$ then $Y \in NPC$

  - To prove $Y \in NPC$ we just need to prove $Y \in NP$ (often easy) and reduce problem in NPC to Y (no lower bound proof needed!).

- Finding the first problem in NPC is somewhat difficult and require quite a lot of formalism

  - The first problem proven to be in NPC was SAT:

    Give a boolean formula, is there an assignment of true and false to the variables that makes the formula true?

  - For example:

    Can $((x_1 \Rightarrow x_2) \vee \neg((\neg x_1 \Leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$ be satisfied?

- Last time we discussed what seems to be a easier problem 3SAT: Given a formula in 3-CNF, is it satisfiable?

  - A formula is in 3-CNF (conjunctive normal form) if it consists of an AND of 'clauses' each of which is the OR of 3 'literals' (a variable or the negation of a variable)
  - Example: $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

- We prove that 3SAT is in NPC, that is, that it is as hard as general SAT.

  - 3SAT $\in NP$
  - SAT $\leq_P$ 3SAT

    (we showed how to transform general formula into 3-CNF in polynomial time.)

3 Clique

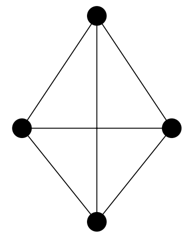
- NP-complete problems arise in many domains

  - Many important graph problems are in NPC.

- Clique: Given a graph $G = (V, E)$ decide if there is a subset $V' \subset V$ of size k such that there is an edge between every pair of vertices in V'

  - Decision version of problem of finding maximal clique.

Example (clique of size 4):

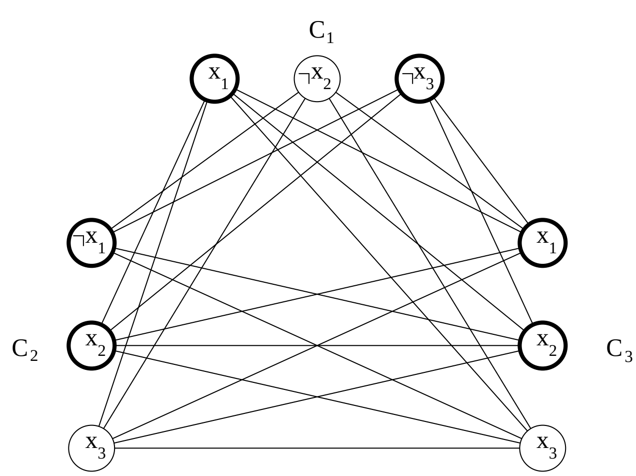


- We could of course solve Clique by testing each of the $\left(\binom{|V|}{k}\right)$ ways of choosing subset of size k.
  - but would take exponential time for $k = \Theta(|V|)$
- Clique is indeed hard:

  $\boxed{\text{Theorem: CLIQUE} \in NPC}$

  Proof:
  - Clique $\in NP$: Given a subset V' we can easily check in polynomial time that $|V'| = k$ and that V' is a clique.
  - 3SAT $\leq_P$ Clique (somewhat surprising since formulas seem to have little to do with graphs):
    * We construct a graph $G = (V, E)$ from a k clause formula $\phi = C_1 \wedge C_2 \wedge C_3 \cdots \wedge C_k$ in 3-CNF:
      For each clause $C_r = (l_1^r \vee l_2^r \vee l_3^r)$ we place triple of vertices v_1^r, v_2^r, v_3^r in V.
      Vertices v_i^r and v_j^s are connected if
      a) $r \neq s$
      b) l_i^r and l_j^s are consistent (not negations of each other)

      Example: $\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



    * Graph can be constructed in polynomial time.

* We have ϕ satisfiable $\Leftrightarrow$ G has clique of size k:

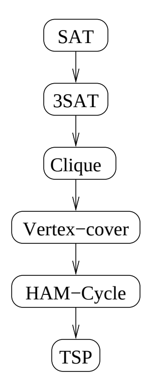
  (Example: ϕ satisfiable by $x_1 = 0, x_2 = 0, x_3 = 1$ and set of white vertices is a clique of size 3.)

  $\Rightarrow$:
  - Each clause C_r contains at least one literal l_i^r assigned 1
  - Each such literal corresponds to vertex v_i^r; pick such a vertex in each clause $\Rightarrow$ k vertices V'
  - For any two vertices $v_i^r, v_j^s \in V'$ ($r \neq s$) both corresponding literals l_i^r and l_i^s are mapped to 1
$\Rightarrow$ they are not complements
$\Rightarrow$ edge in G between v_i^r and v_j^s
$\Rightarrow$ V' clique.

  $\Leftarrow$:
  - Let V' be clique of size k $\Rightarrow$ V' contains exactly one vertex for each triple (no edges between vertices in triple)
  - We can assign 1 to each literal l_i^r corresponding to $v_i^r \in V$ since G contains no edges between inconsistent literals
  - Each clause is satisfiable $\Rightarrow$ ϕ satisfiable.

4 Examples of other problems in NPC

- As mentioned a lot of problems have been proved to be in NPC (and thus we believe them to be hard)
- One example is VERTEX-COVER: Given a graph $G = (V, E)$ decide if there is a set $V' \subset V$ of size k, such that for each edge $e = (u, v) \in E$, $u \in V'$ or $v \in V'$ (or both).
  - Decision version of finding minimal vertex cover.
- We can prove VERTEX-COVER $\in NP$ and CLIQUE $\leq_P$ VERTEX-COVER which means that VERTEX-COVER $\in NPC$.
- We can also prove that VERTEX-COVER$\leq_P$ HAM-CYCLE and we have already discussed that HAM-CYCLE$\leq_P$ TSP, which means that both HAM-CYCLE and TSP are NP-complete.
- We can illustrate our NPC proofs using the following "reduction-graph":



  - As mentioned *many* more problems have been shown NP-complete.

- Even though many important problems are NP-complete, it doesn't mean that we have given up on solving them. Often we are able to solve interesting instances because e.g.
  - they are small (exponential time algorithms work)
  - they are special (solvable in polynomial time)
  - we can find *near optimal* solutions (many so-called *approximation algorithms* have been developed for NPC problems in recent years. For example, its very easy to design algorithm that computes a vertex cover for a graph of size at most twice the minimal cover).