CS420+500: Advanced Algorithm Design and Analysis

Lectures: March 6 + 8, 2017

Prof. Will Evans *Scribe: Amin Aghaee*

In this lecture we:

- Discussed complexity classes ;
- defined NP, NP-Hard;
- NP-Completeness and reduction.

1 NP-Completeness

Firstly, let's start with some definitions:

Definition 1. *Decision Problem: algorithmic questions that can be answered by YES or NO.*

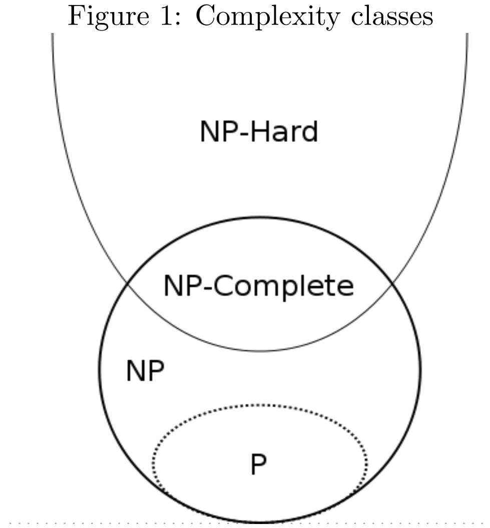
Definition 2. *P: or Polynomial set is set of decision problems decidable in polynomial time. A decision problem L is in P if there exists a polynomial time algorithm A such that $L = \{x | A \text{ accepts } x\}$. (Note that A is a polynomial time algorithm if there exists a positive integer k such that for all inputs x, A halts on input x and either accepts or rejects x in time $O(|x|^k)$.)*

Definition 3. *NP: is set of decision problems that have polynomial time "verifications".*

Definition 4. *Verification: An algorithm V is a polynomial time verifier for a problem L if for every input $x \in L$, there exists a witness w such that V on input (x, w) accepts in time polynomial in $|x|$, and if $x \notin L$, then for all witnesses w, V on input (x, w) rejects in time polynomial in $|x|$.*

Originally, NP comes from non-deterministic polynomial or more precisely from non-deterministic turing machines. Every problem in P is also in NP since the algorithm A for L acts as a verifier that doesn't require a witness.

Figure 1: Complexity classes



1.1 SAT problem

SAT is the set of Boolean formulas in CNF [1] that are satisfiable, that is, there is a truth assignment to the variables in the formula so that the formula evaluates to True.

Theorem 5. *$SAT \in NP$*

Proof: The string w that specifies the truth assignment is a good witness for ϕ. Verifier V needs to only check that w satisfies ϕ (can be done in polynomial time).

The class Co-NP is the set of decision problems L whose complement is in the class NP. The complement of a decision problem L is the set $\{x|x \notin L\}$.

1.2 CLIQUE problem

CLIQUE = $\{< G, k > |G$ is a graph with clique of size $k\}$. Clique of size k has k vertices that all are adjacent to each other.

Theorem 6. *$CLIQUE \in NP$*

witness w for $< G, k >$ is a set of k vertices of G that form a clique. Verifier can check in polynomial time in $| < G, k > |$ that w is a clique or not.

Definition 7. *NP-Hard: set of problems L s.t. if L could be solved in polynomial time, then all other problems in NP could also be solved in polynomial time. Formally, $L \in NP - Hard$ means if $L \in P$ then $L^{'} \in P$ for all $L^{'} \in NP$.*

Definition 8. *NP-Complete: A decision problem L is NP-Complete if:*

1. $L \in NP$
2. $L \in NP - Hard$

Theorem 9 (Cook-Levin 1971)**.** *SAT is NP-complete.*

We are not going to prove theorem 9 in class. NP-Complete contains hardest problems in NP. CLIQUE is an NP-Complete (Yes/No) problem but MAX-CLIQUE is NP-Hard (find maximum size clique in a graph G). Usually, when we convert Yes/No problems to finding problems, they get harder.

2 Reduction to SAT

We know that SAT is NP-Complete problem. It is difficult to prove a problem is NP-hard in the same way that Cook did. However, since we know SAT is NP-hard, we can show that a problem L is NP-hard by showing that a polynomial-time algorithm for L can be used to solve SAT in polynomial time. In other words, by showing how to reduce SAT to L. It is important to do this reduction in a right order.

[1] Conjunctive normal form such as $(x \vee y) \wedge (\overline{x} \vee z)$

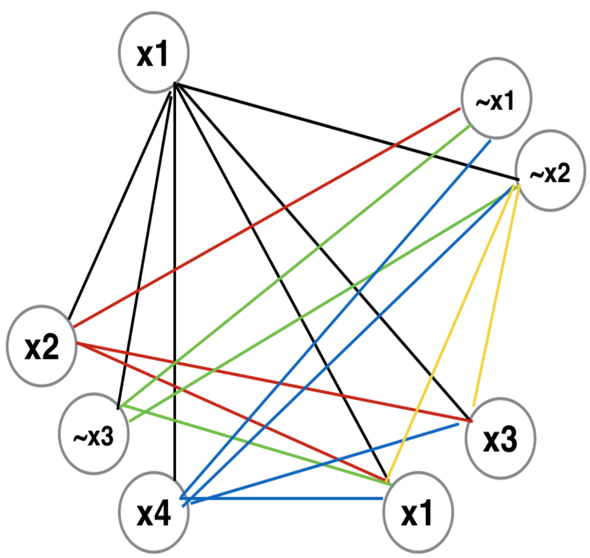
Theorem 10. *CLIQUE* $\in$ *NP-Hard.*
As a proof, we are going to reduce SAT problem to CLIQUE. So we want to transform a formula ϕ *into a graph* $< G, k >$ *so that:*

1. *G contains a clique of size $k \Leftrightarrow \phi$ is satisfiable*
2. *transformation should take polynomial time*

So we do the following three steps:

- Create a vertex for every literal in every clause
- Connect a vertex from i'th clause to j'th clause ($i \neq j$) unless they are negative of each other ($x, \overline{x}$)
- Let k be number of clauses in ϕ

As a example, let say we have $\phi = (x_1) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (x_1 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee x_4)$. Here is a transformed graph:



We claim that $\phi \in$ SAT iff G has k-clique.
($\Rightarrow$) If ϕ has a truth assignment, then every clause has at least one true literal. Thus, we can choose one from each clause of size "k".
($\Leftarrow$) If G has a clique "k" then exactly one vertex from each clause is in ϕ. So we can assign one to each literal vertex and as a result, ϕ is satisfiable.