COMP 409, Lecture 09

February 13, 2003

1 NP-completeness Theory

Let us recall the definition of NP-complete problems and related concepts.

Definition 1 *L is NP-hard if for all $L' \in NP$, $L' \leq_P L$.*

Definition 2 *L is* NP-complete *if $L \in NP$ and $L \in$ NP-hard.*

NP-complete problems can be thought of as those problems that meet both the upper-bound complexity requirement of NP and the lower-bound complexity requirement of NP-hardness. There are two fundamental ideas for NP-complete problems.

1.1 Easily checkable:

A Problem L is in NP if there is a polynomial P and a P-time algorithm A such that for $x \in \Sigma^*$, we have $x \in L$ if and only if there exists $y \in \Sigma^{\leq P(|X|)}$ such that $A(x, y) = 1$ (We can verify a solution to an NP-complete problem in P-time.)

1.2 P-time Reduction ($\leq$)

$L_1 \leq_p L_2$ if there is a P-time function F such that for all $x \in \Sigma^*$ we have that $x \in L_1$ if and only if $x \in L_2$. This relation allows us to compare the complexity of different problems. If we consider the symmetric closure of $\leq_p$, it forms a set of equivalence classes. The NP-complete problems are an equivalence class that contains the hardest problems in NP.

2 P vs. NP

There are uncountably many problems (Σ^* is countable, but 2^{Σ^*} – the set of all subsets of Σ^* – is uncountable.) However, there are countably many NP problems, because NP problems are described by the pair (f:polynomial, A: turing machine) and both are countable. Thus, there are countably many NP-Complete problems.

The $P = NP$ problem has remained open for approximately 30 years now, since Cook introduced it in the early 70's. If we can prove that one problem, $L_1 \in NPC$, is polynomially reducible to a problem $L_2 \in P$, then it follows that $P = NP$. This is because if $L_1 \leq_p L_2$, then that means that for all $L \in NPC$, $L \leq_p L_2$ as well, since $L \leq_p L_1$ (recall that NPC is an equivalence class and thus transitivly closed).

In some of the previous lectures, we proved that $3 - COLOR \leq_p SAT$, and that $SAT \leq_p 3 - COLOR$. We saw that where the first reduction is easy, the second one is tricky. In this sense, we can say that the ease of reduction to a problem is suggestive of its usefulness as a formalism for expressing situations. Some useful statements about P and NP:

1) If $NP \neq P$, then $NP \neq P \bigcup NPC$ (Ladner's Theorem)
2) $SAT \in NPC$
3) If $L \in NPC$ and $L \in P$, then $P = NP$
4) If $NP \neq P$ and $L \in NPC$, then $L \notin P$

3 Clausal Formulas

Definition 3 *Literal : A formula of the form p or of the form $(\neg p)$, where $p \in Prop$.*

Definition 4 *Clause : A disjunction of literals. Can be represented as a set. For example, the clause $p \vee \neg p \vee r$ can be represented as $\{p, \neg q, r\}$. By convention, an empty clause is always false.*

Definition 5 *Clausal formula : Conjunction of clauses, equivalently, a set of clauses. By convention, an empty clausal formula is always true. A clausal formula is said to be in CNF–conjunctive-normal form.*

We can ignore the order of the literals in a clause and the order of clauses in a clausal formula because of the following two lemmas.

Lemma 1 *Commutativity - $(\varphi \wedge \psi) \models (\psi \wedge \varphi)$, and vice-versa.*

Lemma 2 *Associativity - $(\varphi \wedge (\psi \wedge \theta)) \models ((\varphi \wedge \psi) \wedge \theta)$, and vice-versa.*

4 An NP-Complete Problem: Integer Programming

We know that solving a system of equations is simply a problem from linear algebra and that the algorithm for it, namely Gaussian elimination followed by backward substitution, is in P-time - specifically, it can be solved in $O(n^3)$ steps - and thus is in P.

For solving linear inequalities, i.e., Linear Programming (LP), Dantzig in 1945 devised the Simplex algorithm. The simplex method did well in practice;

however, it was shown later to be exponential in the worst case. A $PTIME$ method (Ellipsoid Method, developed in 1978) was then discovered and thus it was proved that $LP \in P$. Unfortunately, this algorithm proved to be terrible in practice. Later, another $PTIME$ method, the Interior-Point Method, was developed, by Karmarkar in 1985. Both the simplex and interior methods are currently in use, the choice depends on the nature of the problem.

Now, restricting the solutions of a linear programming problem to only integer solutions, it would at first seem that the new problem, referred to as Integer Programming (IP), is not very different from the original; however, on closer inspection, we see that IP is not so simple, and that in fact, $IP \in NP{-}Complete$.

To show that IP is NP-complete, first we need to prove that $IP \in NP$. Second, we need to prove that $IP \in$ NP-hard. For this *specific* problem, the first step is hard, but the second step is easy. We will not show that $IP \in NP$. This requires showing that if an IP problem $\boldsymbol{Ax} \leq \boldsymbol{b}$ is feasible, then the number of bits representing the solution is polynomial in the number of bits represensting the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$.

Lemma 3 *$IP \in NP$ because if $\boldsymbol{Ax} = \boldsymbol{b}$ is feasible, then there is a polynomially bounded solution.*

We will show that CNF-SAT$\leq_p$IP. CNF-SAT is satisfiability for formulas of the form

$$\bigwedge_i C_i, \text{ in which } C_i : \bigvee_j \pm p_{ij}$$

where $+p_{ij} = p_{ij}$ and $-p_{ij} = \neg p_{ij}$. It is known that CNF-SAT $\in$ NPC.

Theorem 1 *CNF-SAT $\leq_p$ IP.*

Proof: For each proposition $p_i \in AP(\varphi)$ we introduce an integer variable x_i such that:

1. $x_i = 1$ if p_i is true
2. $x_i = 0$ if p_i is false

So, we have the inequalities: $0 \leq x_i \leq 1$.

How do we convert from clauses to inequalities? To convert a conjunction of clauses in the CNF-SAT problem to an IP problem I_φ, we express each clause as an inequality:

1. p_i is represented by x_i
2. $\neg p_i$ is represented by $(1 - x_i)$
3. $\bigvee l_j$ gives us $\sum u_j \geq 1$, where u_j is x_j if l_j is p_j, and u_j is $1 - x_j$ if l_j is $\neg p_j$.

In this way, each dosjunction is expressed as an inequality. We need to formally show that this conversion can be done in PTIME.

Next, all that is left to be shown is that φ is satisfiable if the coressponding Interger Prgramming problem I_φ is feasible. In other words,

$$\text{models}(\varphi) \neq \emptyset \text{ iff } \text{solutions}(IP_\varphi) \neq \emptyset;$$

However, it is in fact simpler to show the stronger result

$$\text{models}(\varphi) \overset{1-1}{\leftrightarrow} \text{solutions}(IP_\varphi).$$

for this specific reduction.

If for a model τ, $\tau \models \varphi$, a valuation v for IP_φ is given by $v(x_i) = \tau(p_i)$.

Conversely, given a valuation v for IP_φ, a model for φ is given by $\tau(p_i) = v(x_i)$.

Corollary 1 *IP is NP-hard.*

Theorem 2 *IP is NP-complete.*

5 Diophantine Equation

A Diophantine equation is a polynomial equation in which only integer solutions are allowed. Hilbert's 10th Problem asked if a technique for solving a general Diophantine existed. (A set of open problems in mathematics was proposed by Hilbert in 1900.) The impossibility of obtaining a general solution has been proven by Matyasevich (1970).