non-determinism

NP
NP-completeness

COMP3310/3610: Theory of Computation

University of Sydney

Taso Viglas
tasos@it.usyd.edu.au

September 3, 2007

The class $\mathbf{NP}$

- Brute-force search sometimes leads to inefficient algorithms
- Can we always avoid brute-force search?
- $\mathbf{NP}$, a class of problems
- many practical problems of similar complexity do not seem to allow any efficient solution
- (Deterministic) polynomial solutions are not known, but it is also not known whether those problems actually require more than polynomial time

Hamiltonian paths

- given a directed graph $G$ and two of its vertices $s, t$, is there a path connecting $s$ with $t$ that visits all vertices exactly once?
- $H\text{AMPATH} = \{(G, s, t) | G \text{ is a directed graph, with a Hamiltonian path from } s \text{ to } t\}$
- brute-force algorithm will work. check all possible paths from $s$ to $t$
- it is an open problem whether a polynomial solution exists
On $NP$

- many other problems seem to resist polynomial time solutions
- many of these problems are strongly related: if you solve one you have a solution for all of them
- $HAMPATH$: two interesting properties
  - no poly-time algorithm known
  - a correct solution can be verified in poly-time
- in other words, the problem seems hard to solve but easy to verify

Solving versus verifying

- is it easier to solve a problem or just verify the solution to problem?
- this question can be formalized in complexity theoretic terms
- verifiers and poly-time verifiable problems
- $HAMPATH$, $COMPOSITES$
- what about primality? is it poly time verifiable?
- primality is now known to be in $P$

Verifiers and certificates

- consider a problem $A$ and the associated decision problem $x \in A$?
- sometimes it is possible to present proof of membership or a certificate that proves that a given $x \in A$
- (definition 7.18p265) $A$ verifier for a language $A$ is an algorithm $V$ where
  $$A = \{w | V \text{ accepts}(w, c) \text{ for some string } c\}$$
- a polynomial verifier for $A$ runs in time poly. in $w$
- if $A$ has a polynomial verifier, then it is called polynomially verifiable

Verifying certificates

- the verifier uses the information of the certificate $c$
- note that for polynomial verifiers, the certificate must be polynomial in length
- examples of certificates
  - $HAMPATH$: a directed path
  - $COMPOSITES$: a divisor
  - what about primality?
Definitions of \textit{NP}

- (definition 7.19p266) \textit{NP} is the class of languages that have polynomial time verifiers
- remember: \textit{NP} is a polynomial time class
- (theorem 7.20p266) A language is in \textit{NP} iff it is decided by some non-deterministic poly-time Turing machine
- \textit{NP} stands for non-deterministic polynomial
- \textit{P}: languages with poly-time deciders
- \textit{NP}: languages with poly-time verifiers

\textbf{NP and non-determinism}

- (theorem 7.20p266) A language is in \textit{NP} iff it is decided by some non-deterministic poly-time Turing machine
- proof: show how to convert a certificate to a NTM and vice versa
- \[ \Rightarrow \] assume the verifier $V$ runs in time $n^k$
- to decide $w \in A$, on input $w$ (of length $n$
  - non-deterministically select the certificate $c$ of length at most $n^k$
  - run $V$ on $(w, c)$
  - if $V$ accepts, accept, o.w. reject

\textbf{NP and non-determinism}

- (definition 7.21p267) \textit{NTIME}(t(n)) = \{ L | L is a language decided by a $O(t(n))$ NTM \}
- (corollary 7.22p267)
  \[ \textit{NP} = \bigcup_k \textit{NTIME}(n^k) \]
non-determinism

views of non-determinism:
1. several choices, computation tree
2. ability to guess: guess (a poly-length string that defines a path down the tree to an accept) and verify that your guess was correct
3. for input $w$, $\exists c : PolyCheck(c, w)$
   $P$ does not need the quantifier. for input $w$, $PolyCheck(x)$

Examples of problems in $NP$

- $CLIQUE$: given a graph, does it contain a $k$-clique?
- $HAMPATH$: given a graph does it contain a hamiltonian path?
- $SUBSET - SUM$: given a collection of numbers and a target $t$, does there exist a subset with sum equal to $t$?
  $SUBSETSUM = \{(S, t) | S = \{s_1, \ldots, s_k\} and for some \{y_1, \ldots, y_l\} \subseteq S \sum y_i = t\}$
- what about the complements of these sets?
- what kind of certificate can you give for something that does not exist?

Boolean Satisfiability

given a boolean formula $\varphi$ does there exist an assignment of 0,1 to the variables, that makes the entire formula true?
if a satisfying truth assignment exists for $\varphi$ then $\varphi$ is called satisfiable
$SAT = \{\varphi | \varphi$ is a satisfiable boolean formula$\}$
$SAT \in NP$
**Boolean Satisfiability**

- given a boolean formula $\varphi$ does there exist an assignment of 0,1 to the variables, that makes the entire formula true?
- if a satisfying truth assignment exists for $\varphi$ then $\varphi$ is called satisfiable
- $SAT = \{ \varphi | \varphi$ is a satisfiable boolean formula $\}$
- $SAT \in NP$

**Comparing problems and their complexity**

- is $SAT$ more difficult than $CLIQUE$?
- *(theorem 7.32p274)* $3SAT$ is poly-time reducible to $CLIQUE$
- if we can solve $CLIQUE$ then we can solve $3SAT$
- proof: reduce $3SAT$ to $CLIQUE$

**Reducibility**

- recall the notion of mapping-reducibility, $A \leq_m B$
- *(definition 5.20p207)* $A$ is mapping reducible to $B$ written $A \leq_m B$ if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that for all $w$
  $$w \in A \iff f(w) \in B$$
  The function $f$ is called the reduction from $A$ to $B$
- extend the definition for poly-time mapping reductions

**Poly-time reducibility**

- *(definition 7.28p272)* a function $f : \Sigma^* \rightarrow \Sigma^*$ is a poly-time computable function if some polynomial time TM on input $w$ halts with $f(w)$ on its tape
- *(definition 7.29p272)* $A$ is poly-time mapping reducible to $B$ written $A \leq_P B$ if there is a poly-time computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that for all $w$
  $$w \in A \iff f(w) \in B$$
  The function $f$ is called the polynomial time reduction from $A$ to $B$
- this is also called polynomial time many-one reducibility
NP-completeness

(definition 7.34p276) A language $B$ is $NP$-complete if it satisfies two conditions:
1. $B \in NP$
2. all $A$ in $NP$ polynomially reduce to $B$: $\forall A \in NP : A \leq_P B$

$NP$-complete is a class of languages so we can also say or write $B \in NP$-complete

NP-completeness was introduced independently by Cook and Levin (70’s)
the first $NP$-complete problem, $SAT$
the Cook-Levin theorem: $SAT$ is $NP$-complete (theorem 7.27p267)
proof: use computation histories: construct a formula such that an accepting computation exists iff the formula is satisfiable
show how to reduce any $A \in NP$ to a $SAT$ question
all we know is that $A \in NP$ and therefore the only thing we can use is that a NTM computation exists

3SAT

(corollary 7.42p282) $3SAT$ is $NP$-complete
proof: any $SAT$ formula can be converted to a $3SAT$ that is up to polynomially longer
use the distributive laws of boolean algebra to write the Cook-Levin reduction formula as a cnf
$
\varphi_{cell} \land \varphi_{start} \land \varphi_{move} \land \varphi_{accept}$
big clauses (more than 3 literals) can be split into smaller ones by introducing new variables
Clique

- A clique is a complete graph.
- The clique problem: given a graph $G$ and a number $k$, does $G$ contain a sub-graph of $k$ nodes that form a clique?
- $CLIQUE$ is $NP$-complete
  - $CLIQUE$ is in $NP$ and
  - $3SAT$ reduces to $CLIQUE$, $3SAT \leq_p CLIQUE$

Vertex cover

- Given a graph $G$, a vertex cover is a subset of vertices such that every edge in $G$ connects to a vertex in the vertex cover.
- Decision version: does $G$ contain a vertex cover of size $k$?
- (Theorem 7.44p284) $VERTEXCOVER$ is $NP$-complete
- Reduce from $3SAT$

Vertex cover

(Theorem 7.44p284) $VERTEXCOVER$ is $NP$-complete
- Reduce from $3SAT$

Hamiltonian paths

- Given a directed graph $G$ and two of its vertices $s, t$, is there a path connecting $s$ with $t$ that visits all vertices exactly once?
- $HAMPATH = \{(G, s, t)|G$ is a directed graph, with a Hamiltonian path from $s$ to $t\}$
- (Theorem 7.46p286) $HAMPATH$ is $NP$-complete
- Reduce from $3SAT$
(theorem 7.55 p291) \textit{UHAMPATH} is \textit{NP}-complete

- reduce the directed version to the undirected