CS 420: Advanced Algorithm Design and Analysis
Spring 2015 – Lecture 19

Department of Computer Science
University of British Columbia

March 17, 2015

Announcements

Assignments...

- Asst6/7...(due March 19)

Midterm III...

- Q/A session...March 24; 5:30-7:00; DMPT 110
- Exam...March 25; 5:30-7:00; DMPT 110
- ...on *all* course material up to and including March 19 lecture

Announcements (cont.)

Readings...

- matchings and network flows [Kleinberg&Tardos, Chapt. 7], [Cormen et al., Chapt. 26], [Dasgupta et al., Chapter 7]
- reductions and NP-hardness [Kleinberg&Tardos, Chapt. 8, 11], [Cormen et al., Chapt. 34,35]

Last classes...

Matchings and Network Flows

- network flows
  - definitions
  - relationship with bipartite matchings
  - duality

Last classes...

Reductions and relative hardness of problems

- reductions...treated more formally
- overview of problems with efficient algorithms
  ... and related problems with no known efficient algorithm
- the complexity classes **P** and **NP**
- **NP**-hardness and **NP**-completeness

Last classes...

Reductions and relative hardness of problems

- some examples of reductions establishing **NP**-hardness and **NP**-completeness
  - HC $\leqslant_{\mathrm{P}}$ TSP
  - Clique $\leqslant_{\mathrm{P}}$ LargestCommonSubgraph
  - VC $\leqslant_{\mathrm{P}}$ DominatingSet
  - 3-SAT $\leqslant_{\mathrm{P}}$ VC

A review of some graph problems

left column

-spanning trees
  min cost
  maximum width

-path problems
  min-cost
  min colour-transitions
  Eulerian path

-graph colouring
  2-colouring (bipartite)
  4-colouring (planar graph)

A review of some graph problems

left column	**right column**
-spanning trees	
min-cost	bounded-degree MST
maximum width	bounded-diameter MST
-path problems	
min-cost	longest (simple) path
min colour-transitions	min total colours
Eulerian path	Hamiltonian path
-graph colouring	
2-colouring (bipartite)	3-colourability
4-colouring (planar graph)	3-colouring (planar graph)

A review of some graph problems

left column

-matchings etc.
    max size (bipartite)
    vertex cover (bipartite)
    general (non-bipartite)
    b-matchings

-network flows etc.
    max value
    integral/general capacities
    vertex capacities
    minimum cut
    edge/vertex-disjoint paths

A review of some graph problems

left column

-matchings etc.
- max size (bipartite)
- vertex cover (bipartite)
- general (non-bipartite)
- *b*-matchings

-network flows etc.
- max value
- integral/general capacities
- vertex capacities
- minimum cut
- edge/vertex-disjoint paths

right column

- 3-d matching (triangle cover)
- maximum independent set
- vertex cover (tripartite)

- flows with edge costs
- undirected flows with lower bounds
- vertex-disjoint connecting paths

A review of some graph problems

All of the **left column** problems have *efficient* solutions: their decision versions belong to the complexity class **P**, defined to be the family of decision problems (languages) that can be decided (recognized) in time bounded by some polynomial in the input size.

Why are we interested in *polynomial time*?

- generous definition of tractable
- often equates to tractable in practice
- closure properties (composition)
- invariance under natural computation models

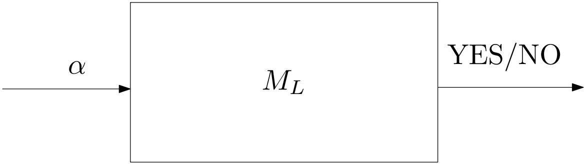
A review of some graph problems

None of the **right column** problems are known to have *efficient* solutions.

Nevertheless, their decision versions all admit efficient *certification*; i.e. a short proof/certificate that the answer is YES. They all belong to the complexity class **NP** is defined to be the family of decision problems (languages) whose membership can be certified/verified in time bounded by some polynomial in the input size.

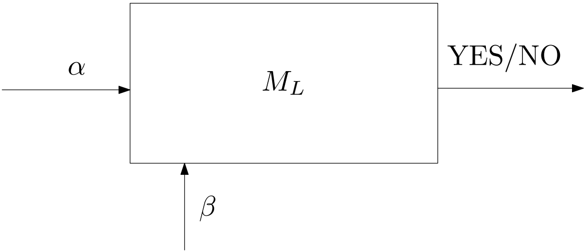
NP stands for *non-deterministic* polynomial-time: certification corresponds to acceptance by a non-deterministic machine.

Deterministic language acceptance



Machine M_L *accepts* L if:
$\alpha \in L$ if and only if M_L outputs YES on input α

Non-deterministic language acceptance



Machine M_L *non-deterministically accepts* L if:
$\alpha \in L$ if and only if there exists a string β such that M_L outputs YES on input (α, β).

The complexity classes **P** and **NP**

P denotes the set of languages that can be (deterministically) accepted in time bounded by some polynomial in the input length.
NP denotes the set of languages that can be (non-deterministically) accepted in time bounded by some polynomial in the input length.

The complexity classes **P** and **NP**

It turns out that all of the **right column** problems are as hard as any problem in **NP**, up to polynomial factors, which is abbreviated **NP**-hard. Since they are also in **NP** they belong to the class **NP**-complete.

NP-complete problems have the property that they have polynomial-time solutions (i.e. they belong to **P**) if and only if **P**=**NP**, i.e. all problems in **NP** have polynomial-time solutions.

The complexity classes **P** and **NP**

How could we possibly show that some problem X is **NP**-hard? We don't even know all of the problems in **NP**!

- it is straightforward once we know some **NP**-hard problem A: simply demonstrate A $\lesssim_{t(n)}$X, where $t(n)$ is some polynomial in n. (Hereafter, we write A $\lesssim_{\mathrm{P}}$X)
- the real breakthrough was the demonstration of a *first* **NP**-hard problem

Today...

Reductions and relative hardness of problems

- The Cook-Levin theorem: establishing the first NP-hard problem
- more examples of reductions establishing **NP**-hardness and **NP**-completeness

Review of Propositional Logic...

Boolean Expressions

A Boolean expression over the set of Boolean variables $\{x_1, x_2, \ldots, x_n\}$ is defined (recursively) as:

1. a variable
2. the negation of a Boolean expression
3. the disjunction (or) of two Boolean expressions
4. the conjunction (and) of two logical expressions

A Boolean expression is *satisfiable* if there is an assignment of truth values to its variables such the the expression *evaluates* to `true`.

Review of Propositional Logic...

Conjunctive Normal Form

By applying DeMorgan's distributive laws, any Boolean expression can be converted to an equivalent expression E in *conjunctive normal form* (CNF):

$$E \equiv D_1 \wedge D_2 \wedge \ldots \wedge D_t$$

a conjunction of *disjuncts*, where each disjunct D_i has the form

$$L_{i,1} \vee L_{i,2} \vee \ldots \vee L_{i,s}$$

a disjunction of *literals*, where each literal $L_{i,j}$ is either a variable x_j or the negation of a variable $\overline{x}_i$.

Review of Propositional Logic...

Conjunctive Normal Form

A formula in k-CNF has the property that each of its disjuncts has at most k literals. For example:

$$(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_1 \vee x_2) \wedge (\overline{x}_2 \vee x_4) \wedge (\overline{x}_1 \vee x_4)$$

is a Boolean expression in 2-CNF.

Review of Propositional Logic...

Satisfiability

The language SAT is defined as the set of all satisfiable Boolean expressions. Its restriction k-SAT is the set of all satisfiable Boolean expressions in k-CNF.

Note:

- 2-SAT is in **P** , since 2-SAT $\leq_{\mathrm{P}}$digraph_connectivity
- k-SAT $\leq_{\mathrm{P}}$SAT and SAT $\leq_{\mathrm{P}}$3-SAT
- SAT is in **NP**

Cook's Theorem...

Theorem: SAT is also **NP**-hard

Literally ... for every language L in **NP**, $L \leqslant_{\mathrm{P}} \mathrm{SAT}$

How could this be proved?

- A language L is in **NP** iff there is a non-deterministic machine M that accepts strings $\alpha \in L$ in $|\alpha|^k$ time, for some fixed k
- So, it suffices to show how to construct a Boolean expression $E(\alpha)$ that says "there exists a string β such that the pair (α, β) is accepted by M in at most $|\alpha|^k$ steps".

The Vertex Cover Problem

Recall that the VERTEX-COVER problem takes as input a graph G and an integer k and asks if G has a vertex cover of size k, i.e. a subset $V_c \subseteq V$ such that every edge in E has at least one endpoint in V_c.

Note

- For *bipartite graphs* the vertex cover problem is in **P**
- In general VERTEX-COVER is in **NP**

The Vertex Cover Problem

In fact...
Theorem: VERTEX-COVER is **NP**-complete.

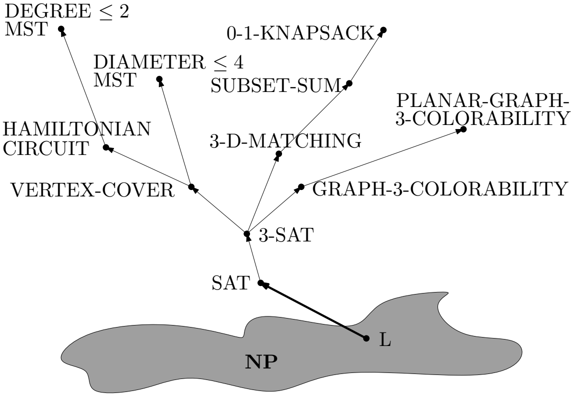
Proof:
We show that 3-SAT $\leq_{\mathrm{P}}$VERTEX-COVER

Today...

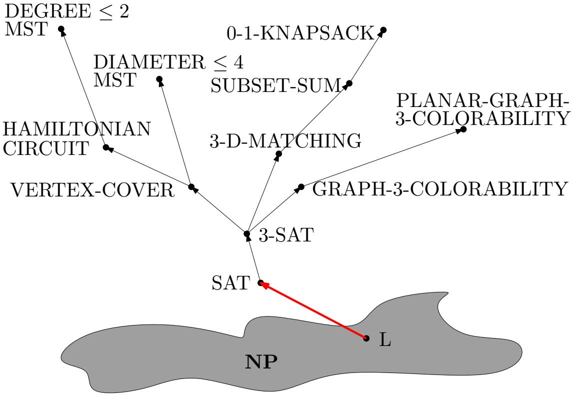
Building the **NP**-hardness reduction tree

- overview of reductions
- some selected examples
  - 3-SAT $\leq_{\mathrm{P}}$ 3-D-MATCHING
  - 3-D-MATCHING $\leq_{\mathrm{P}}$ SUBSET-SUM

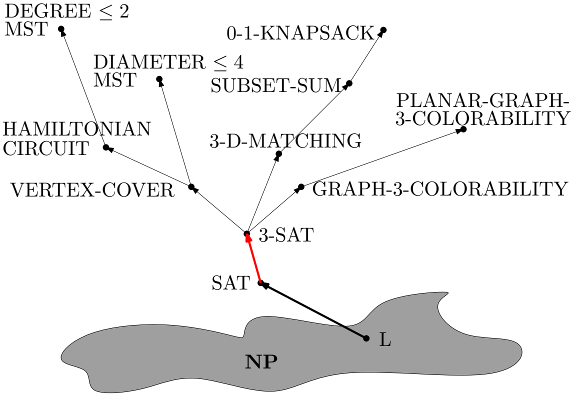
Overview of Reductions



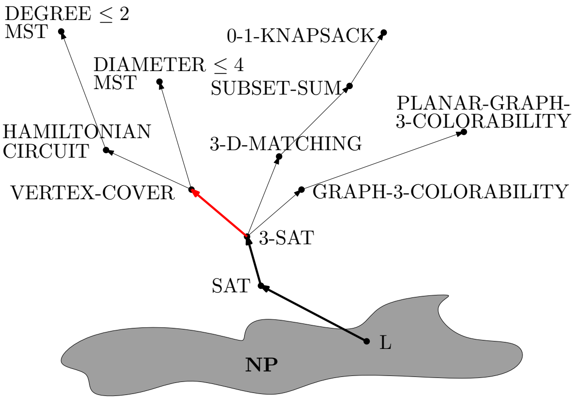
Overview of Reductions



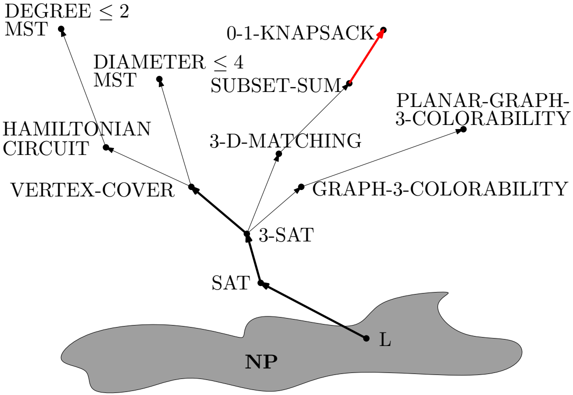
Overview of Reductions



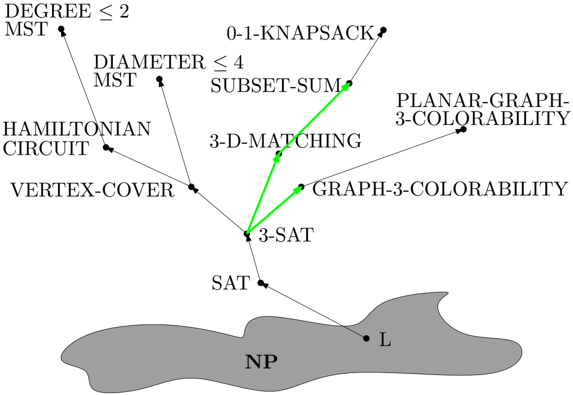
Overview of Reductions



Overview of Reductions



Overview of Reductions



Some more examples

- 3-SAT $\leqslant_{\mathrm{P}}$ 3-D-MATCHING
- 3-D-MATCHING $\leqslant_{\mathrm{P}}$ SUBSET-SUM