NPTEL MOOC,JAN-FEB 2015
Week 8, Module 7

DESIGN AND ANALYSIS OF ALGORITHMS

Intractability: P and NP

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Checking algorithms

- Checking algorithm C for problem
- Takes in an input instance I and a solution "certificate" S for I
- C outputs yes if S represents a valid solution for I, no otherwise

The class NP

- Checking algorithm C that verifies a solution S for input instance I runs in time polynomial in size(I)
- Factorization, satisfiability, travelling salesman, vertex cover, independent set, … are all in NP
  - If we convert an optimization problem to a checking problem by providing a bound, we add only a log factor for binary search through solution space

Why “NP”

- Non-deterministic Polynomial time
- “Guess” a solution and check it
- Origins in computability theory
  - Non-deterministic Turing machines …

P and NP

- P is the class of problems with regular polynomial time algorithms (worst-case complexity)
- P is included in NP — generate a solution and check it!
- Is P = NP?
  - Is efficient checking same as efficient generation?
  - Intuitively this should not be the case

$P \neq NP$?

- A more formal reason to believe this?
- Many "natural" problems are in NP
  - Factorization, satisfiability, travelling salesman, vertex cover, independent set, …
- These are all inter-reducible
  - Like vertex cover, independent set
- If we can solve one efficiently, we can solve them all!

Boolean satisfiability

* Boolean variables x,y,z,…
* Clause — disjunction of literals, (x || !y || z || …. || w)
* Formula — conjunction of clauses, C & D & … & E
* 3-SAT — each clause has at most 3 literals

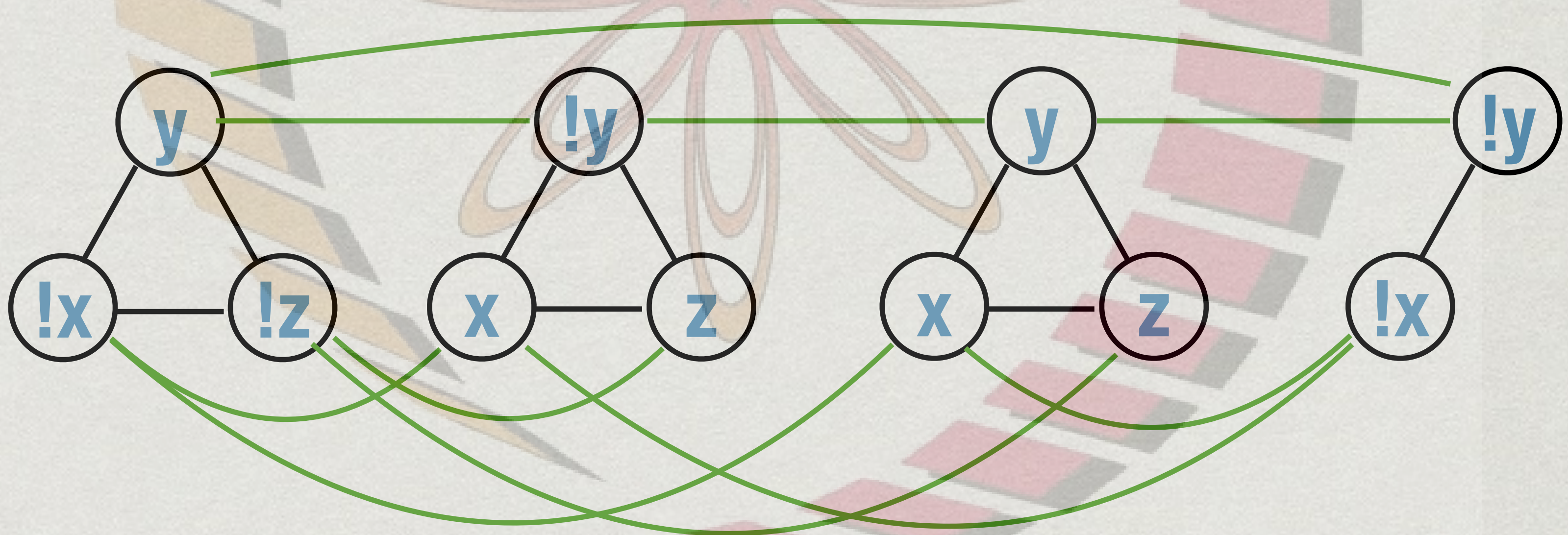
Reducing SAT to 3-SAT

- Consider a 5 literal clause (v|| !w || x || !y || z)
- Introduce a new literal and split the clause
  - (v|| !w || a) & (!a || x || !y || z)
  - This formula is satisfiable iff original clause is
- Repeat till all clauses are of size 3 or less
  - (v|| !w || a) & (!a || x || b) & (!b || !y || z)
- If SAT is hard, so is 3-SAT

3-SAT to independent set

- Construct a graph from a 3-SAT formula

  (!x || y || !z) & (x || !y || z) & (x || y || z) & (!x || !y)



- Independent set picks one literal per clause to satisfy
  - Edges enforce consistency across clauses
- Ask for size of independent set = number of clauses

Reductions within NP

- SAT → 3-SAT, 3-SAT → independent set, independent set ↔ vertex cover
- Reduction is transitive, so SAT → vertex cover, …
- Other inter-reducible NP problems
  - Travelling salesman, integer linear programming …
- All these problems are "equally" hard

NP-Completeness

Cook-Levin Theorem

- Every problem in NP can be reduced to SAT
- Original proof is by encoding computations of Turing machines
- Can replace by encoding of any “generic” computation model — boolean circuits, register machines …

NP-Completeness

- SAT is said to be complete for NP
  - It belongs to NP
  - Every problem in NP reduces to it
- Since SAT reduces to 3-SAT, 3-SAT is also NP-complete
- In general, to show P is NP-complete, reduce some existing NP-complete problem to P

P ≠ NP?

- A large class of practically useful problems are NP-complete
  - Scheduling, bin-packing, optimal tours …
- If one of them has a solution in P, all of them do
- Many smart people have been working on these problems for centuries
  - Empirical evidence that NP is different from P
  - But a formal proof is elusive, and worth $1 million!