COT 6936: Topics in Algorithms

Giri Narasimhan

ECS 254A / EC 2443; Phone: x3748

giri@cs.fiu.edu

https://moodle.cis.fiu.edu/v2.1/course/view.php?id=612

What are *NP-Complete* problems?

- These are the hardest problems in *NP*.
- A problem p is *NP-Complete* if
  - there is a polynomial-time reduction from **every** problem in *NP* to p.
  - p $\in$ *NP*
- How to prove that a problem is *NP-Complete*?

- **Cook's Theorem**: [1972]
  - The SAT problem is *NP-Complete*.

Steve Cook, Richard Karp, Leonid Levin

The SAT Problem: an example

- Consider the boolean expression:

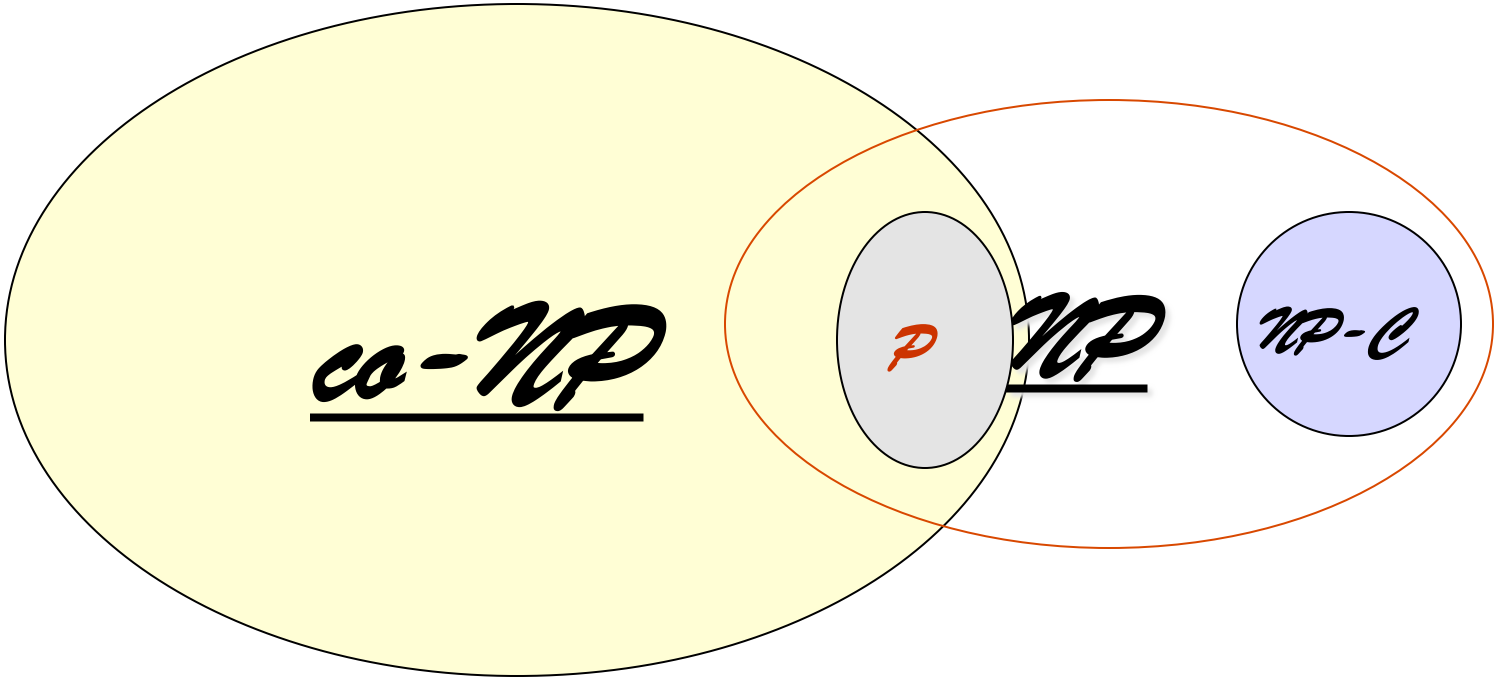
  $C = (a \vee \neg b \vee c) \wedge (\neg a \vee d \vee \neg e) \wedge (a \vee \neg d \vee \neg c)$
- Is C satisfiable? [Does there exist a True/False assignments to the boolean variables a, b, c, d, e, such that C is True?]
- If there are n boolean variables, then there are 2^n different truth value assignments.
- However, a solution can be quickly verified!

The SAT (Satisfiability) Problem

- Input: Boolean expression C in Conjunctive normal form (CNF) in n variables and m clauses.
- Question: Is C satisfiable?
  - Let $C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$
  - Where each $C_i = \left(y_1^i \vee y_2^i \vee \cdots \vee y_{k_i}^i\right)$
  - And each $y_j^i \in \{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_n, \neg x_n\}$
  - We want to know if there exists a truth assignment to all the variables in the boolean expression C that makes it true.
- Steve Cook showed that the problem of deciding whether a non-deterministic Turing machine T accepts an input w or not can be written as a boolean expression C_T for a SAT problem. The boolean expression will have length bounded by a polynomial in the size of T and w.

> - How to now prove Cook's theorem? Is SAT in $\mathcal{NP}$?
> - Can every problem in $\mathcal{NP}$ be poly. reduced to it ?

The problem classes and their relationships



More *NP-Complete* problems

3SAT

- **Input**: Boolean expression C in Conjunctive normal form (CNF) in n variables and m clauses. Each clause has at most three literals.
- **Question**: Is C satisfiable?
  - Let $C = C_1 \wedge C_2 \wedge \dots \wedge C_m$
  - Where each $C_i = (y_1^i \vee y_2^i \vee y_3^i)$
  - And each $y_j^i \in \{x_1, \neg x_1, x_2, \neg x_2, \dots, x_n, \neg x_n\}$
  - We want to know if there exists a truth assignment to all the variables in the boolean expression C that makes it true.

3SAT is *NP-Complete*.

3SAT is *NP-Complete*

- 3SAT is in *NP*.
- SAT can be reduced in polynomial time to 3SAT.
- This implies that every problem in *NP* can be reduced in polynomial time to 3SAT. Therefore, 3SAT is *NP-Complete*.
- So, we have to design an algorithm such that:
  - Input: an instance C of SAT
  - Output: an instance C' of 3SAT such that satisfiability is retained. In other words, C is satisfiable if and only if C' is satisfiable.

3SAT is *NP-Complete*

- Let C be a SAT instance with clauses $C_1, C_2, \ldots, C_m$
- Let C_i be a disjunction of $k > 3$ literals.

  $C_i = \quad y_1 \vee y_2 \vee \ldots \vee y_k$
- Rewrite C_i as follows:

  $C'_i = \quad (y_1 \vee y_2 \vee z_1) \wedge$

  $(\neg z_1 \vee y_3 \vee z_2) \wedge$

  $(\neg z_2 \vee y_4 \vee z_3) \wedge$

  …

  $(\neg z_{k-3} \vee y_{k-1} \vee y_k)$
- Claim: C_i is satisfiable if and only if C'_i is satisfiable.

More *NP-Complete* problems?

2SAT

- **Input**: Boolean expression C in Conjunctive normal form (CNF) in n variables and m clauses. Each clause has at most three literals.
- **Question**: Is C satisfiable?
  - Let $C = C_1 \wedge C_2 \wedge \dots \wedge C_m$
  - Where each $C_i = \left(y_1^i \vee y_2^i\right)$
  - And each $y_j^i \in \{x_1, \neg x_1, x_2, \neg x_2, \dots, x_n, \neg x_n\}$
  - We want to know if there exists a truth assignment to all the variables in the boolean expression C that makes it true.

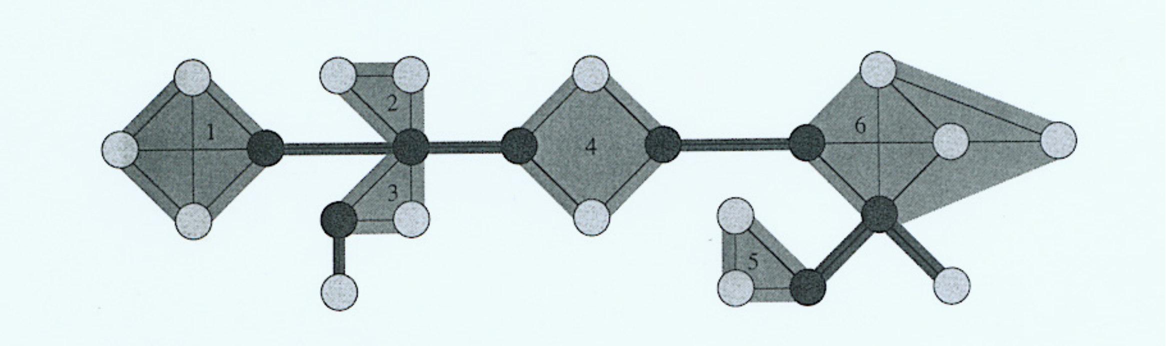
2SAT is in *P*.

2SAT is in $\mathcal{P}$

- If there is only one literal in a clause, it must be set to true.
- If there are two literals in some clause, and if one of them is set to false, then the other must be set to true.
- Using these constraints, it is possible to check if there is some inconsistency.
- How? Homework: do not submit!

The CLIQUE Problem

- A **clique** is a completely connected subgraph.

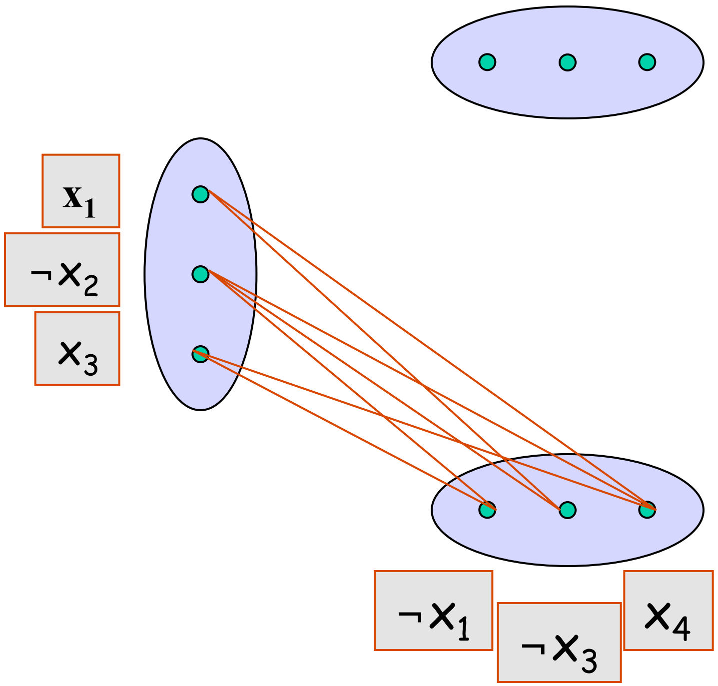


CLIQUE

- **Input**: Graph G(V,E) and integer k
- **Question**: Does G have a clique of size k?

CLIQUE is *NP-Complete*

- CLIQUE is in *NP*.
- Reduce 3SAT to CLIQUE in polynomial time.
- $F = (x_1 \vee \neg x_2 \vee x_3)\ (\neg x_1 \vee \neg x_3 \vee x_4)\ (x_2 \vee x_3 \vee \neg x_4)\ (\neg x_1 \vee \neg x_2 \vee x_3)$

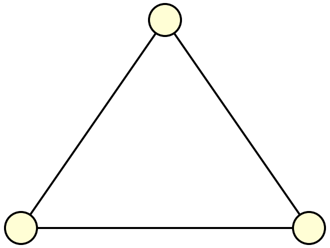


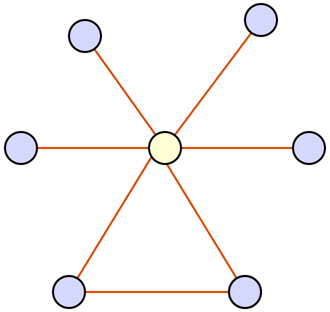
F is satisfiable if and only if G has a clique of size k where k is the number of clauses in F.

Vertex Cover

A vertex cover is a set of vertices that "covers" all the edges of the graph.

Examples



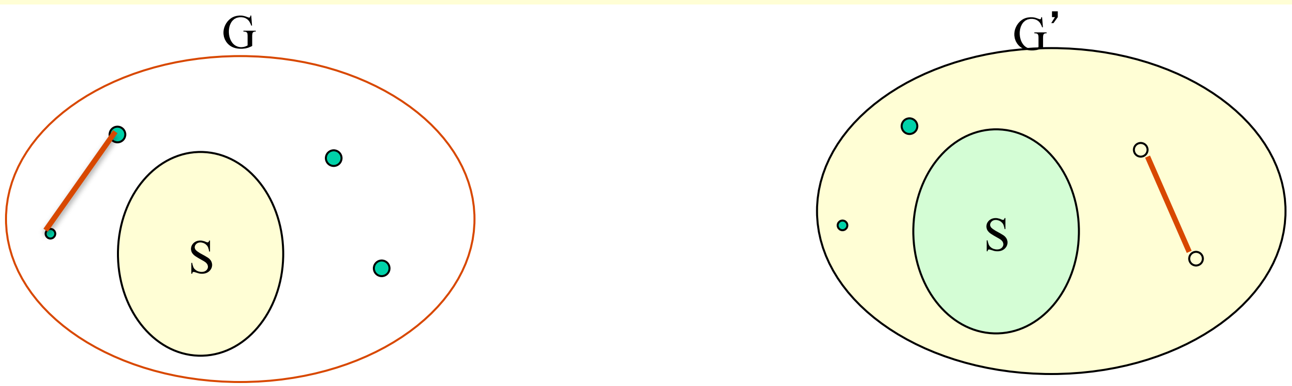


Vertex Cover (VC)

Input: Graph G, integer k

Question: Does G contain a vertex cover of size k?

- VC is in *NP*.
- polynomial-time reduction from CLIQUE to VC.
- Thus VC is *NP-Complete.*



Claim: G' has a clique of size k' if and only if G has a VC of size k = n - k'

Hamiltonian Cycle Problem (HCP)

Input: Graph G

Question: Does G contain a hamiltonian cycle?

- HCP is in *NP*.
- There exists a polynomial-time reduction from 3SAT to HCP.
- Thus HCP is *NP-Complete.*

Shortest Path vs Longest Path

Input: Graph G with edge weights, vertices u and v, bound B

Question: Does G contain a shortest path from u to v of length at most B?

Question: Does G contain a longest path from u to v of length at most B?

Homework: Listen to Cool MP3:

http://www.cs.princeton.edu/~wayne/kleinberg-tardos/longest-path.mp3

Perfect (2-D) Matching vs 3-D Matching

1. **Input**: Bipartite graph, G(U,V,E)
   Question: Does G have a perfect matching?

2. **Input**: Sets U and V, and E = subset of U×V
   Question: Is there a subset of E of size |U| that covers U and V? [Related to 1.]

3. **Input**: Sets U,V,W, & E = subset of U×V×W
   Question: Is there a subset of E of size |U| that covers U, V and W?

Coping with NP-Completeness

- Approximation: Search for an "almost" optimal solution with provable quality.
- Randomization: Design algorithms that find "provably" good solutions with high prob and/or run fast on the average.
- Restrict the inputs (e.g., planar graphs), or fix some input parameters.
- Heuristics: Design algorithms that work "reasonably well".

Reading

- Read Background
  - Algorithms & Discrete Math Fundamentals
    - Cormen, et al., Chapters 1-16, 22-25
  - NP-Completeness
    - Cormen et al., Chapter 34
    - Appendix (p187-288) form Garey & Johnson
- Next Class
  - Approximation Algorithms
    - Cormen et al., Chapter 35
    - Kleinberg, Tardos, Chapter 11
    - Books by Vazirani and Hochbaum/Shmoys

Required Reading for Feb 6

- Network Flow
  - Ford Fulkerson Algorithm
- Linear Programming
  - Standard LP
  - Dual LP
  - Feasibility and feasible region