TSP: Traveling Salesperson Problem

- **Input:**
  - Weighted graph, *G*
  - Length bound, B
- **Question:**
  - Is there a traveling salesperson tour in *G* of length at most B?
- Is TSP in $\mathcal{NP}$?
  - YES. Easy to verify a given solution.
- Is TSP in $\mathcal{P}$?
  - OPEN!
  - One of the greatest unsolved problems of this century!
  - Same as asking: **Is $\mathcal{P} = \mathcal{NP}$?**

Terminology (Cont'd)

- Complexity Class $\mathcal{P}$:
  - Set of all problems *p* for which polynomial-time algorithms exist.
- Complexity Class $\mathcal{NP}$:
  - Set of all problems *p* for which polynomial-time verification algorithms exist.
- Complexity Class *co-*$\mathcal{NP}$:
  - Set of all problems *p* for which polynomial-time verification algorithms exist for their **complements**, I.e., their complements are in $\mathcal{NP}$.

Terminology (Cont'd)

- **Reductions:** $p_1 \rightarrow p_2$
  - A problem p_1 is reducible to p_2, if there exists an algorithm R that takes an instance i_1 of p_1 and outputs an instance i_2 of p_2, with the constraint that the solution for i_1 is YES if and only if the solution for i_2 is YES.
  - Thus, R converts YES (NO) instances of p_1 to YES (NO) instances of p_2.
- Polynomial-time reductions: $p_1 \xrightarrow{P} p_2$
  - Reductions that run in polynomial time.

- If $p_1 \xrightarrow{P} p_2$, then
  - If p_2 is easy, then so is p_1. $p_2 \in \mathcal{P} \Rightarrow p_1 \in \mathcal{P}$
  - If p_1 is hard, then so is p_2. $p_1 \notin \mathcal{P} \Rightarrow p_2 \notin \mathcal{P}$

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

NP-Complete vs *NP-Hard*

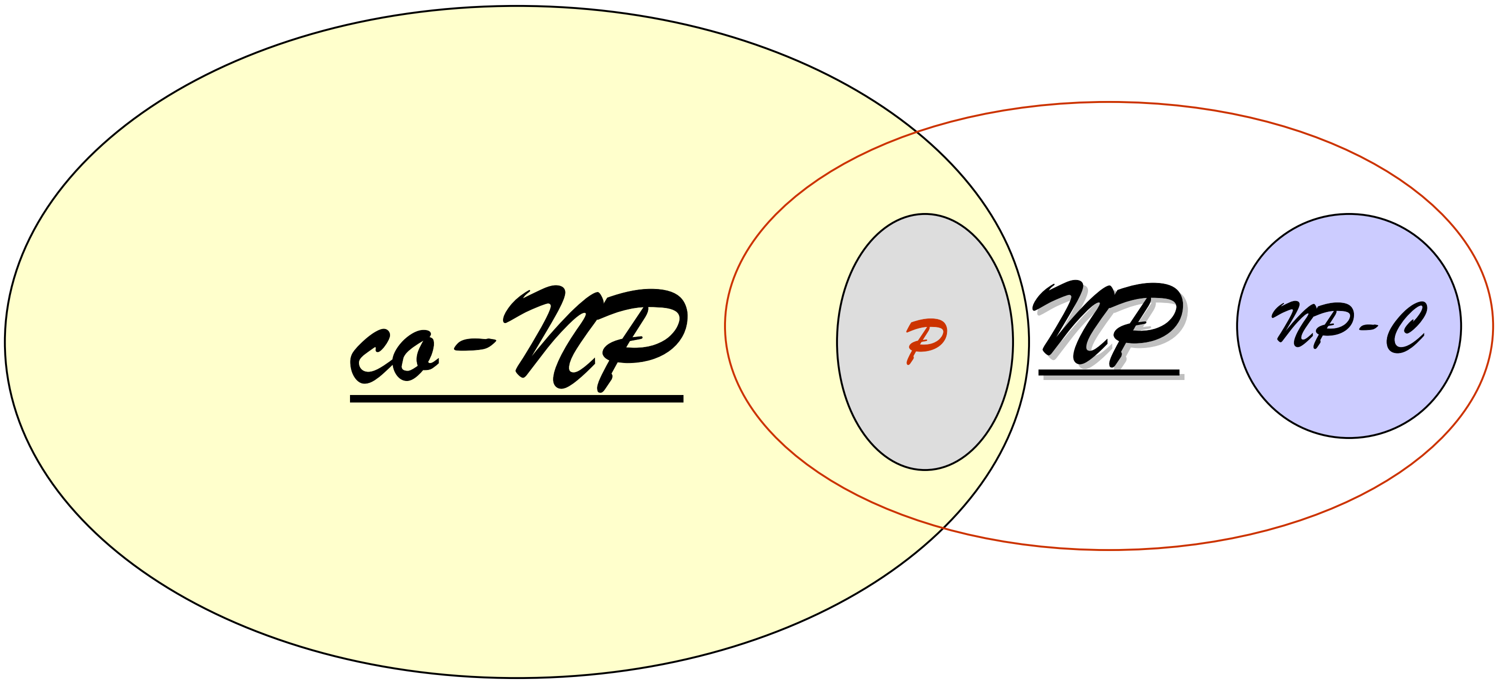
- A problem p is *NP-Complete* if
  - there is a polynomial-time reduction from **every** problem in *NP* to p.
  - p $\in$ *NP*
- A problem p is *NP-Hard* if
  - there is a polynomial-time reduction from **every** problem in *NP* to p.

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 $C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$
  - Where each $C_i = \left(y_1^i \vee y_2^i \vee y_3^i\right)$
  - 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.

3SAT is *NP-Complete.*

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 $C = C_1 \wedge C_2 \wedge \ldots \wedge C_m$
  - Where each $C_i = \left(y_1^i \vee y_2^i\right)$
  - And each $y_j^i \in \{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_n, \neg x_n\}$
  - We want to know if there exists a truth assignment to all the variables in the boolean expression C that makes it true.

2SAT is in *P*.

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 an instance of SAT 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:

  $$C'_i = \quad (y_1 \vee y_2 \vee z_1) \wedge$$
  $$(\neg z_1 \vee y_3 \vee z_2) \wedge$$
  $$(\neg z_2 \vee y_4 \vee z_3) \wedge$$
  $$\ldots$$
  $$(\neg z_{k-3} \vee y_{k-1} \vee y_k)$$

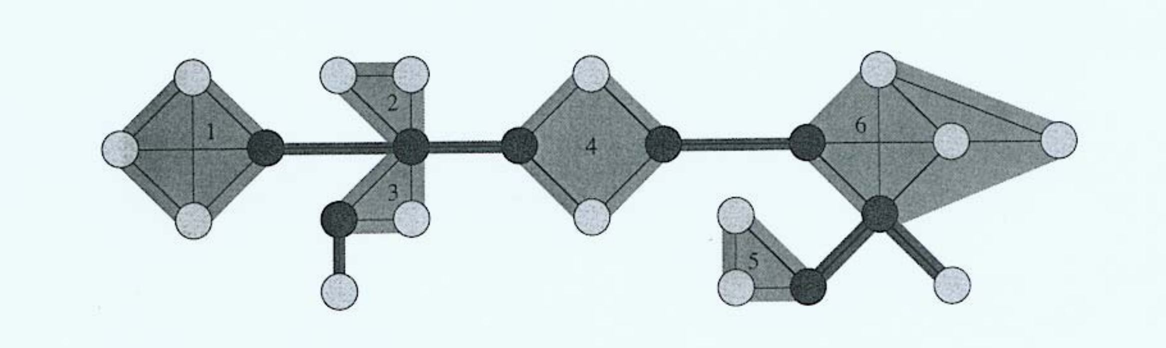
- Claim: C_i is satisfiable if and only if C'_i is satisfiable.

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 problem!

The CLIQUE Problem

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

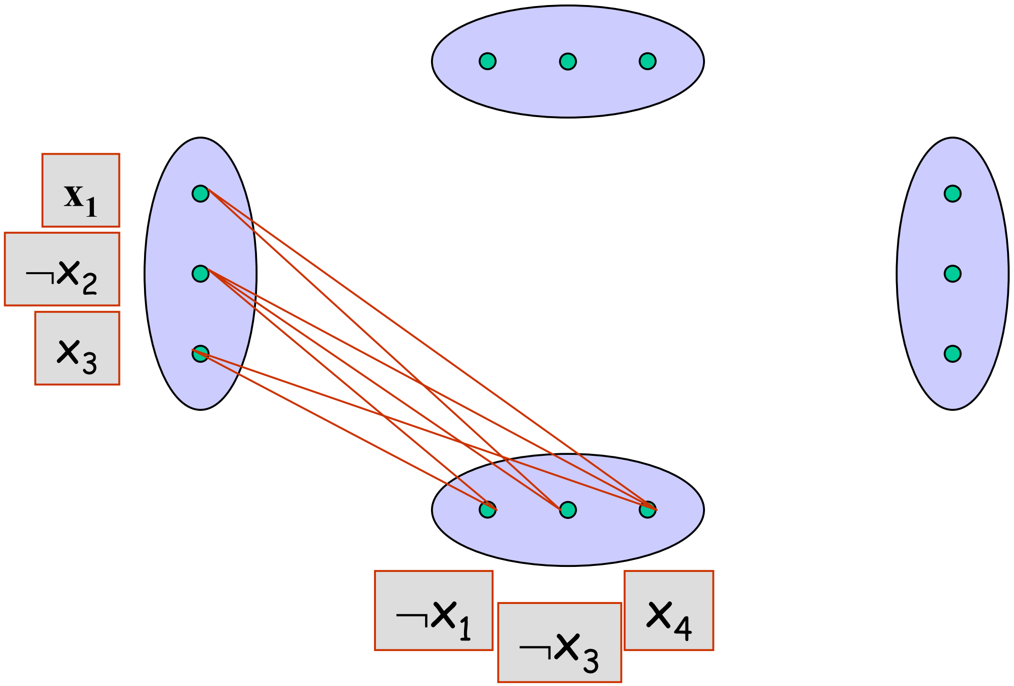


CLIQUE

- 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.
- $F = (x_1 \vee \neg x_2 \vee x_3)\ (\neg x_1 \vee \neg x_3 \vee x_4)\ (x_2 \vee x_3 \vee \neg x_4)\ (\neg x_1 \vee \neg x_2 \vee x_3)$

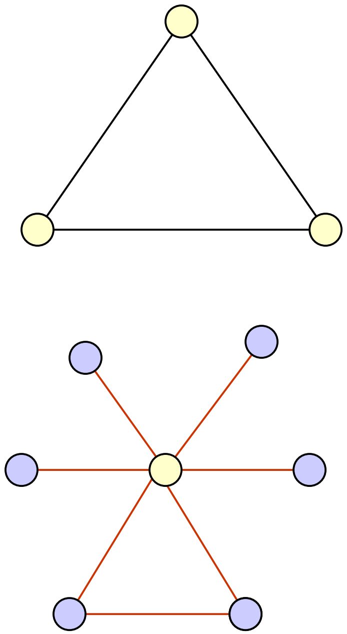


CLAIM: F is satisfiable
if and only if
G has a clique of size k
where k is the number of
Clauses in F.

Vertex Cover

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

Examples

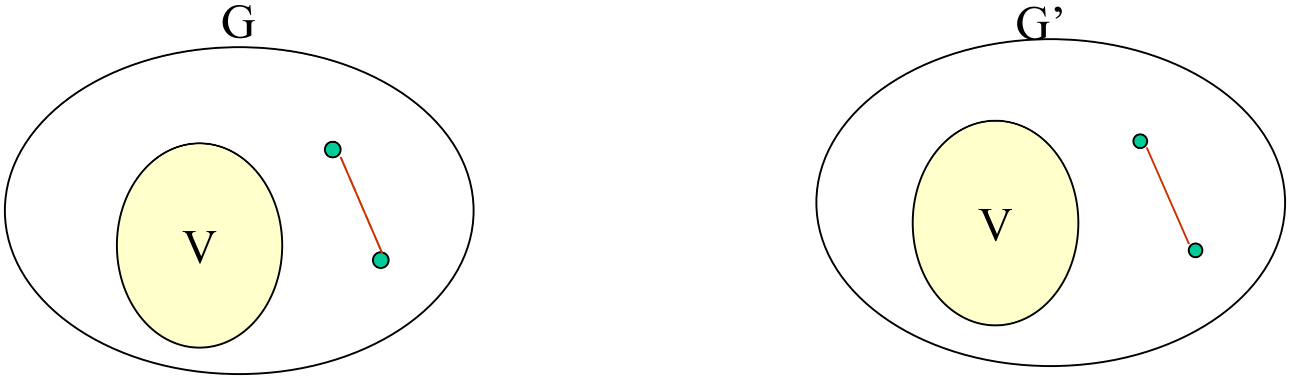


Vertex Cover (VC)

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.*

- Notes/animations by Yi Ge!