Two Notions

- Let $R \subseteq \Sigma^* \times \Sigma^*$ be a binary relation on strings.
- R is called **polynomially decidable** if
$$\{x; y : (x, y) \in R\}$$
is in P.
- R is said to be **polynomially balanced** if $(x, y) \in R$ implies $|y| \leq |x|^k$ for some $k \geq 1$.

An Alternative Characterization of NP

Proposition 28 (Edmonds, 1965) *Let $L \subseteq \Sigma^*$ be a language. Then $L \in NP$ if and only if there is a polynomially decidable and polynomially balanced relation R such that*

$$L = \{x : (x, y) \in R \textit{ for some } y\}.$$

- Suppose such an R exists.
- L can be decided by this NTM:
  - On input x, the NTM guesses a y of length $\leq |x|^k$ and tests if $(x, y) \in R$ in polynomial time.
  - It returns "yes" if the test is positive.

The Proof (continued)

- Suppose that $L \in$ NP.
- NTM N decides L in time $|x|^k$.
- Define R as follows: $(x, y) \in R$ if and only if y is the encoding of an accepting computation of N on input x.
- Clearly R is polynomially balanced because N is polynomially bounded.
- R is also polynomially decidable because it can be efficiently verified by simulation.
- Finally $L = \{x : (x, y) \in R \text{ for some } y\}$ because N decides L.

Comments

- Any "yes" instance x of an NP problem has at least one **succinct certificate** or **polynomial witness** y of its being a "yes" instance.
- "No" instances have none.
- Certificates are short and easy to verify.
  - An alleged satisfying truth assignment for SAT, an alleged Hamiltonian path for HAMILTONIAN PATH.
- Certificates may be hard to generate (otherwise, NP equals P), but verification must be easy.
- NP is the class of *easy-to-verify* problems.

You Have an NP-Complete Problem (for Your Thesis)

- From Propositions 23 (p. 163) and Proposition 24 (p. 164), it is the least likely to be in P.
- Approximations.
- Special cases.
- Average performance.
- Randomized algorithms.
- Exponential-time algorithms that work well for small problems.
- "Heuristics" (and pray).

3SAT

- kSAT, where $k \in \mathbb{Z}^+$, is the special case of SAT.
- The formula is in CNF and all clauses have *exactly* k literals (repetition of literals is allowed).
- For example,

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3).$$

3SAT Is NP-Complete

- Recall Cook's Theorem (p. 177) and the reduction of CIRCUIT SAT to SAT (p. 156).
- The resulting CNF has at most 3 literals for each clause.
  - This shows that 3SAT where each clause has at most 3 literals is NP-complete.
- Finally, duplicate one literal once or twice to make it a 3SAT formula.

Another Variant of 3SAT

Proposition 29 3SAT *is NP-complete for expressions in which each variable is restricted to appear at most three times, and each literal at most twice.*

- 3SAT here requires only that each clause has *at most* 3 literals.
- Consider a 3SAT expression in which x appears k times.

The Proof (continued)

- Replace the first occurrence of x by x_1, the second by x_2, and so on, where $x_1, x_2, \ldots, x_k$ are k *new* variables.
- Add $(\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \land \cdots \land (\neg x_k \lor x_1)$ to the expression $(x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k \Rightarrow x_1)$.
  - Each clause may have fewer than 3 clauses.
- The equivalent expression satisfies the condition for x.

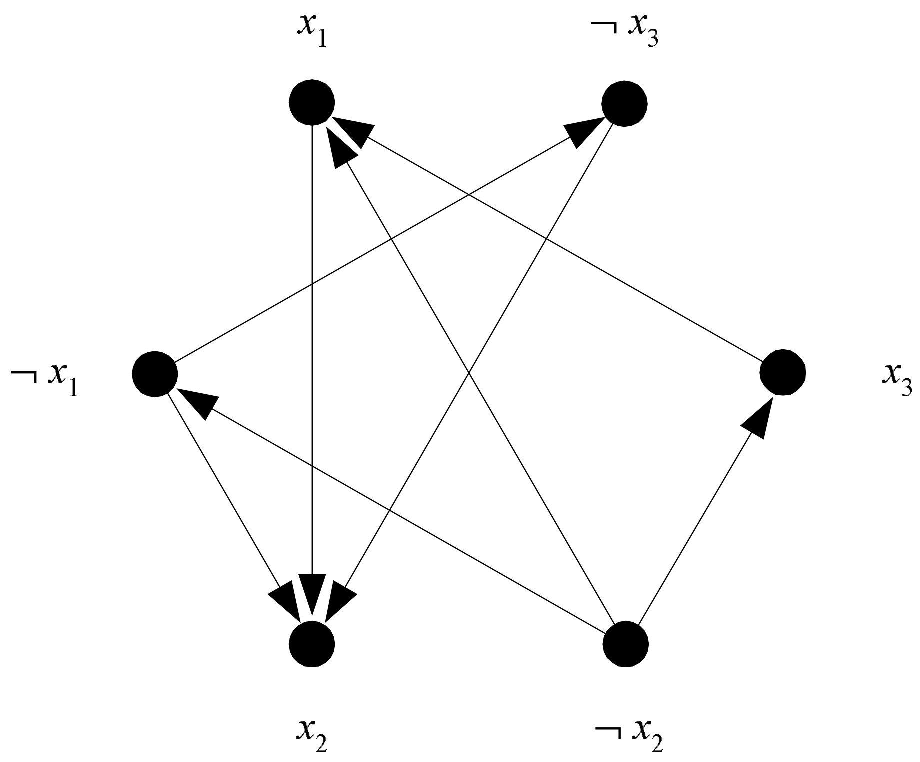
2SAT and Graphs

- Let ϕ be an instance of 2SAT, in which each clause has exactly 2 literals.
- Define graph $G(\phi)$ as follows:
  - The nodes are the variables and their negations.
  - Add edges $(\neg\alpha, \beta)$ and $(\neg\beta, \alpha)$ to $G(\phi)$ if $\alpha \vee \beta$ is a clause in ϕ.
    * For example, if $x \vee \neg y \in \phi$, add $(\neg x, \neg y)$ and (y, x).
    * *Two* edges are added for each clause.
  - Think of the edges as $\neg\alpha \Rightarrow \beta$ and $\neg\beta \Rightarrow \alpha$.
  - b is reachable from a iff $\neg a$ is reachable from $\neg b$.
  - Paths in $G(\phi)$ are valid implications.

Illustration

Digraph for

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_2 \vee x_3).$$



Properties of $G(\phi)$

Theorem 30 *ϕ is unsatisfiable if and only if there is a variable x such that there are paths from x to $\neg x$ and from $\neg x$ to x in $G(\phi)$.*

- Suppose that such paths exist, but ϕ can be satisfied by a truth assignment T.
  - Without loss of generality, assume $T(x) = \texttt{true}$.
  - As there is a path from x to $\neg x$ and $T(\neg x) = \texttt{false}$, there must be an edge (α, β) on this path such that $T(\alpha) = \texttt{true}$ and $T(\beta) = \texttt{false}$.
  - Hence $(\neg\alpha \vee \beta)$ is a clause of ϕ.
  - But this clause is *not* satisfied by T, a contradiction.

The Proof (continued)

- Suppose there is no variable with such paths in $G(\phi)$.
- We shall construct a satisfying truth assignment.
- It is enough that no edges go from true to false.
- Pick any node α which has not had a truth value and there is no path from it to $\neg\alpha$ (always doable by assumption, why?).
- Assign nodes reachable from α true and their negations false.
  - The negations are those nodes that can reach $\neg\alpha$.

The Proof (continued)

- The above steps are well-defined.
  - If α could reach both β and $\neg\beta$, then there would be a path from $\neg\beta$ to $\neg\alpha$, hence a path from α to $\neg\alpha$!
  - If there were a path from α to a node y already assigned false, then $\neg y$ can reach $\neg\alpha$ and α has been assigned false before!
- We keep picking such α's until we run out of them.
- Every node must have had a truth value.
  - If α does not, it must be because there is a path from it to $\neg\alpha$, but then the algorithm could have picked $\neg\alpha$!
- The assignments make sure a false never follows a true.

2SAT Is in NL $\subseteq$ P

- By Corollary 21 on p. 145, coNL equals NL.
- We need to show only that recognizing unsatisfiable expressions is in NL.
- In nondeterministic logarithmic space, we can test the conditions of Theorem 30 by guessing a variable x and testing if $\neg x$ is reachable from x *and* if $\neg x$ can reach x.
  - See the algorithm for REACHABILITY (p. 70).

Generalized 2SAT: MAX2SAT

- Consider a CNF in which all clauses have two literals.
- Let $K \in \mathbb{N}$.
- MAX2SAT is the problem of whether there is a truth assignment that satisfies at least K of the clauses.
- MAX2SAT becomes 2SAT when K equals the number of clauses.
- MAX2SAT is an optimization problem.
- MAX2SAT is in NP: Guess a truth assignment and verify the count.

MAX2SAT Is NP-Complete[a]

- Consider the following 10 clauses:

$$(x) \wedge (y) \wedge (z) \wedge (w)$$
$$(\neg x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge (\neg z \vee \neg x)$$
$$(x \vee \neg w) \wedge (y \vee \neg w) \wedge (z \vee \neg w)$$

- Let the 2SAT formula $r(x, y, z, w)$ represent the conjunction of these clauses.
- How many clauses can we satisfy?
- The clauses are symmetric with respect to x, y, and z.

[a] Garey, Johnson, Stockmeyer, 1976.

The Proof (continued)

All of x, y, z **are true:** By setting w to true, we *can* satisfy $4 + 0 + 3 = 7$ clauses.

Two of x, y, z **are true:** By setting w to true, we *can* satisfy $3 + 2 + 2 = 7$ clauses; by setting w to false, we *can* satisfy $2 + 2 + 3 = 7$ clauses.

One of x, y, z **is true:** By setting w to false, we *can* satisfy $1 + 3 + 3 = 7$ clauses, whereas by setting w to true, we *can* satisfy only $2 + 3 + 1 = 6$ clauses.

None of x, y, z **is true:** By setting w to false, we *can* satisfy $0 + 3 + 3 = 6$ clauses, whereas by setting w to true, we *can* satisfy only $1 + 3 + 0 = 4$ clauses.

The Proof (continued)

- Any truth assignment that satisfies $x \lor y \lor z$ can be extended to satisfy 7 of the 10 clauses and no more.
- The remaining truth assignment can be extended to satisfy only 6 of them.
- The reduction from 3SAT ϕ to MAX2SAT $R(\phi)$:
  - For each clause $C_i = (\alpha \lor \beta \lor \gamma)$ of ϕ, add **group** $r(\alpha, \beta, \gamma, w_i)$ to $R(\phi)$.
  - If ϕ has m clauses, then $R(\phi)$ has $10m$ groups.
- Set $K = 7m$.

The Proof (continued)

- We now show that K clauses of $R(\phi)$ can be satisfied if and only if ϕ is satisfiable.
- Suppose $7m$ clauses of $R(\phi)$ can be satisfied.
  - 7 clauses must be satisfied in each group because each group can only have at most 7 clauses satisfied.
  - But all clauses *in* ϕ must be satisfied.
- Suppose all clauses of ϕ are satisfied.
  - Each group can set its w_i appropriately to have 7 clauses satisfied.

NAESAT

- The NAESAT (for "not-all-equal" SAT) is like 3SAT.
- But we require additionally that there be a satisfying truth assignment under which no clauses have the three literals equal in truth value.
  - Each clause must have one literal assigned true and one literal assigned false.

NAESAT Is NP-Complete[a]

- Recall the reduction of CIRCUIT SAT to SAT on p. 156.
- It produced a CNF ϕ in which each clause has at most 3 literals.
- Add the same variable z to all clauses with fewer than 3 literals to make it a 3SAT formula.
- We will argue that the new formula $\phi(z)$ is NAE-satisfiable if and only if the original circuit is satisfiable.

[a] Karp, 1972.

The Proof (continued)

- Suppose T NAE-satisfies $\phi(z)$.
  - $\bar{T}$ also NAE-satisfies $\phi(z)$.
  - Under either T or $\bar{T}$, variable z takes the value false.
  - This truth assignment must satisfy all clauses of ϕ.
  - So it satisfies the original circuit.

The Proof (continued)

- Suppose there is a truth assignment that satisfies the circuit.
  - Then there is a truth assignment T that satisfies every clause of ϕ.
  - Extend T by adding $T(z) = \texttt{false}$ to obtain T'.
  - T' satisfies $\phi(z)$.
  - So in no clauses are all three literals false under T'.
  - Under T', in no clauses are all three literals true.
    * Review the construction on p. 157 and p. 158.

Undirected Graphs

- An **undirected graph** $G = (V, E)$ has a finite set of nodes, V, and a set of *undirected* edges, E.
- It is like a graph except that the edges have no directions and there are no self-loops.
- We use $[i, j]$ to denote the fact that there is an edge between node i and node j.

Independent Sets

- Let $G = (V, E)$ be an undirected graph.
- $I \subseteq V$.
- I is **independent** if whenever $i, j \in I$, there is no edge between i and j.
- The INDEPENDENT SET problem is this: Given an undirected graph and a goal K, is there an independent set of size K?
  - Many applications.

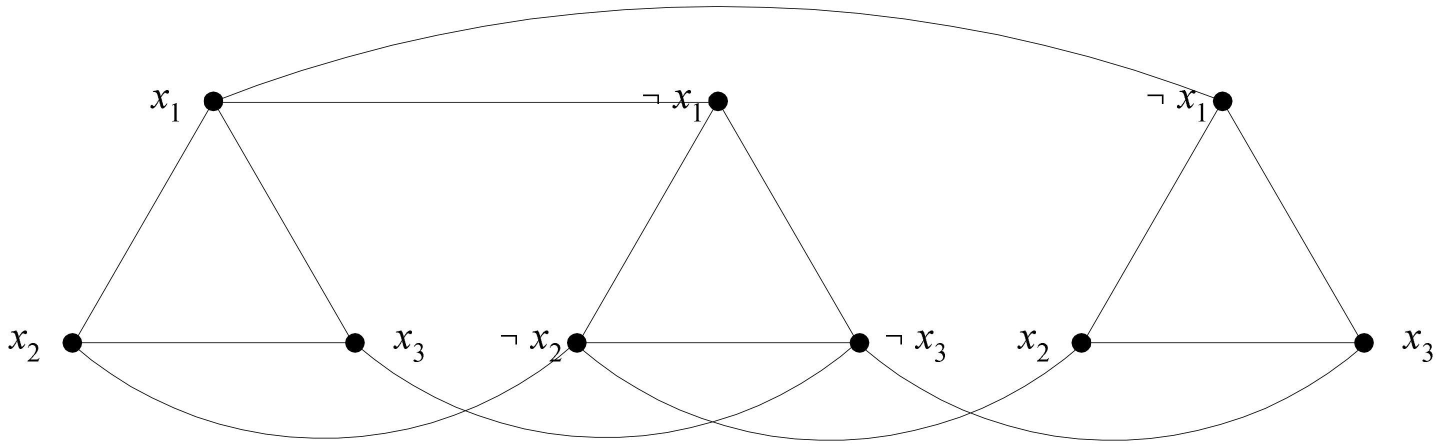
INDEPENDENT SET Is NP-Complete

- This problem is in NP: Guess a set of nodes and verify that it is independent and meets the count.
- If a graph contains a triangle, any independent set can contain at most one node of the triangle.
- We consider graphs whose nodes can be partitioned in m disjoint triangles.
  - If the subproblem is hard, the original problem is at least as hard.

Reduction from 3SAT to INDEPENDENT SET

- Let ϕ be an instance of 3SAT with m clauses.
- We will construct graph G (with constraints as said) with $K = m$ such that ϕ is satisfiable if and only if G has an independent set of size K.
- There is a triangle for each clause with the literals as the nodes.
- Add additional edges between x and $\neg x$ for every variable x.

A Sample Construction



$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3).$$

The Proof (continued)

- Suppose G has an independent set I of size $K = m$.
  - An independent set can contain at most m nodes, one from each triangle.
  - An independent set of size m exists if and only if it contains exactly one node from each triangle.
  - Truth assignment T assigns true to those literals in I.
  - T is consistent because contradictory literals are connected by an edge, hence not both in I.
  - T satisfies ϕ because it has a node from every triangle, thus satisfying every clause.

The Proof (continued)

- Suppose a satisfying truth assignment T exists for ϕ.
  - Collect one node from each triangle whose literal is true under T.
  - This set of m nodes must be independent by construction.

Corollary 31 4-DEGREE[a] INDEPENDENT SET *is NP-complete.*

Theorem 32 INDEPENDENT SET *is NP-complete for planar graphs.*

---

[a]The degrees in the graph are at most 4 if we start with NAESAT.

CLIQUE and NODE COVER

- We are given an undirected graph G and a goal K.
- CLIQUE asks if there is a set of K nodes that form a **clique**, which have all possible edges between them.
- NODE COVER asks if there is a set C with K or fewer nodes such that each edge of G has at least one of its endpoints in C.

Both CLIQUE and NODE COVER Are NP-Complete

Corollary 33 CLIQUE *is NP-complete.*

- Let $\bar{G}$ be the **complement** of G, where $[x, y] \in \bar{G}$ if and only if $[x, y] \notin G$.
- Then I is a clique in G if and only if I is an independent set in $\bar{G}$.

Corollary 34 NODE COVER *is NP-complete.*

- I is an independent set of $G = (V, E)$ if and only if $V - I$ is a node cover of G.

MIN CUT and MAX CUT

- A **cut** in an undirected graph $G = (V, E)$ is a partition of the nodes into two nonempty sets S and $V - S$.
- The size of a cut $(S, V - S)$ is the number of edges between S and $V - S$.
- MIN CUT is in P.
- MAX CUT asks if there is a cut of size at least K.