Outline

• the class NP
  – NP-complete problems: NAE-3-SAT, max-cut
• the class co-NP
• the class NP ∩ coNP
Not-All-Equal 3SAT

\[(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_3) \land \ldots \land (\ldots)\]

**Theorem:** the following language is NP-complete:

\[\text{NAE3SAT} = \{ \varphi : \text{\varphi is a 3-CNF formula for which there exists a truth assignment in which every clause has at least 1 true literal and at least 1 false literal} \} \]

• Proof:
  – Part 1: NAE3SAT ∈ NP. Proof?
  – Part 2: NAE3SAT is NP-hard. Reduce from?
NAE3SAT is NP-complete

• We are reducing from the language:
  CIRCUIT-SAT = \{C : C is a Boolean circuit for which there exists a satisfying truth assignment\}

to the language:

  NAE3SAT = \{φ : φ is a 3-CNF formula for which there exists a truth assignment in which every clause has at least 1 true literal and at least 1 false literal\}
NAE3SAT is NP-complete

- Recall reduction to 3SAT
  - variables $x_1, x_2, ..., x_n$, gates $g_1, g_2, ..., g_m$
  - produce clauses:

  $\bigvee_{\ell \in \{1, 2\}} g_i$
  $z_1 \quad z_2$

  $\bigwedge_{\ell \in \{1, 2\}} g_i$
  $z_1 \quad z_2$

  • $(\neg z_1 \lor g_i)$
  • $(\neg z_2 \lor g_i)$
  • $(\neg g_i \lor z_1 \lor z_2)$

  $\neg g_i$
  $z$

  • $(g_i \lor z)$
  • $(\neg z \lor \neg g_i)$

  Output gate $g_m$:
  • $(g_m)$

  not all true in a satisfying assignment
NAE3SAT is NP-complete

• modified reduction to NAE3SAT
  – variables $x_1, x_2, ..., x_n$, gates $g_1, g_2, ..., g_m$
  – produce clauses:

  \[
  \begin{align*}
  \lor g_i & \quad \bullet (-z_1 \lor g_i \lor w) \\
  z_1 & \quad \bullet (-z_2 \lor g_i \lor w) \\
  z_2 & \quad \bullet (-z_1 \lor z_2 \lor z_2) \\
  \land g_i & \quad \bullet (-g_i \lor z_1 \lor w) \\
  z_1 & \quad \bullet (-g_i \lor z_2 \lor w) \\
  z_2 & \quad \bullet (-z_1 \lor -z_2 \lor g_i) \\
  \end{align*}
  \]

output gate $g_m$:

  \[
  \begin{align*}
  & \bullet (g_m \lor w) \\
  \end{align*}
  \]
NAE3SAT is NP-complete

• Does the reduction run in polynomial time?

• YES maps to YES
  – already know how to get a satisfying assignment to the BLUE variables
  – set $w = \text{FALSE}$

\[
\begin{align*}
& (\neg z_1 \lor g_i \lor w) \\
& (\neg z_2 \lor g_i \lor w) \\
& (\neg g_i \lor z_1 \lor z_2) \\
& (\neg g_i \lor z_1 \lor w) \\
& (\neg g_i \lor z_2 \lor w) \\
& (\neg z_1 \lor \neg z_2 \lor g_i) \\
& (g_i \lor z \lor w) \\
& (\neg z \lor \neg g_i \lor w) \\
& (g_m \lor w)
\end{align*}
\]
NAE3SAT is NP-complete

• NO maps to NO
  – given NAE assignment A
  – complement A’ is a NAE assignment
  – A or A’ has $w = \text{FALSE}$
  – must have TRUE BLUE variable in every clause
  – we know this implies C satisfiable

\[
\begin{align*}
&\neg z_1 \lor g_i \lor w \\
&\neg z_2 \lor g_i \lor w \\
&\neg g_i \lor z_1 \lor z_2 \\
&\neg g_i \lor z_1 \lor w \\
&\neg g_i \lor z_2 \lor w \\
&\neg z_1 \lor \neg z_2 \lor g_i \\
&g_i \lor z \lor w \\
&\neg z \lor \neg g_i \lor w \\
&g_m \lor w
\end{align*}
\]
MAX CUT

- Given graph $G = (V, E)$
  - a cut is a subset $S \subset V$
  - an edge $(x, y)$ crosses the cut if $x \in S$ and $y \in V - S$ or $x \in V - S$ and $y \in S$
  - search problem:
    
    find cut maximizing number of edges crossing the cut
MAX CUT

• Given graph $G = (V, E)$
  – a cut is a subset $S \subset V$
  – an edge $(x, y)$ crosses the cut if $x \in S$ and $y \in V - S$
    or $x \in V - S$ and $y \in S$
  – search problem:
    find cut maximizing number of edges crossing the cut
**Theorem:** the following language is NP-complete:

\[
\text{MAX CUT} = \{(G = (V, E), k) : \text{there is a cut } S \subseteq V \text{ with at least } k \text{ edges crossing it}\}
\]

- **Proof:**
  - Part 1: \(\text{MAX CUT} \in \text{NP}. \) Proof?
  - Part 2: \(\text{MAX CUT} \) is NP-hard.
    - reduce from?
MAX CUT is NP-complete

• We are reducing from the language:

\[ \text{NAE3SAT} = \{ \varphi : \varphi \text{ is a 3-CNF formula for which there exists a truth assignment in which every clause has at least 1 true literal and at least 1 false literal} \} \]

to the language:

\[ \text{MAX CUT} = \{(G = (V, E), k) : \text{there is a cut } S \subseteq V \text{ with at least } k \text{ edges crossing it}\} \]
MAX CUT is NP-complete

• The reduction:
  – given instance of NAE3SAT (n nodes, m clauses):
    $$(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_5) \land \ldots \land (\neg x_2 \lor x_3 \lor x_3)$$
  – produce graph $G = (V, E)$ with node for each literal

• triangle for each 3-clause
• parallel edges for each 2-clause
MAX CUT is NP-complete

- if cut selects TRUE literals, each clause contributes 2 if NAE, and < 2 otherwise
- need to penalize cuts that correspond to inconsistent truth assignments
- add $n_i$ parallel edges from $x_i$ to $\neg x_i$ ($n_i = \#$ occurrences)
  (repeat variable in 2-clause to make 3-clause for this calculation)
MAX CUT is NP-complete

- triangle for each 3-clause
- parallel edges for each 2-clause
- \(n_i\) parallel edges from \(x_i\) to \(\neg x_i\)
- set \(k = 5m\)

- YES maps to YES
  - take cut to be TRUE literals in a NAE truth assignment
  - contribution from clause gadgets: 2m
  - contribution from \((x_i, \neg x_i)\) parallel edges: 3m
MAX CUT is NP-complete

- triangle for each 3-clause
- parallel edges for each 2-clause
- \( n_i \) parallel edges from \( x_i \) to \( \neg x_i \)
- set \( k = 5m \)

• NO maps to NO

  - **Claim**: if cut has \( x_i, \neg x_i \) on same side, then can move one to opposite side without decreasing # edges crossing cut
    - contribution from \((x_i, \neg x_i)\) parallel edges: 3m
    - contribution from clause gadgets must be 2m
    - conclude: there is a NAE assignment
**Claim**: if cut has $x_i$, $\neg x_i$ on same side, then can move one to opposite side without decreasing # edges crossing cut

- **Proof:**

$$a + n_i \geq a + b \text{ or } b + n_i \geq a + b$$
coNP

• Is NP closed under complement?

Can we transform this machine:

\( x \in L \)  \( x \notin L \)

\( q_{\text{reject}} \)  \( q_{\text{accept}} \)  \( q_{\text{accept}} \)  \( q_{\text{reject}} \)

into this machine?
coNP

- language L is in coNP iff its complement (co-L) is in NP

- it is believed that NP ≠ coNP

- note: P = NP implies NP = coNP
  - proving NP ≠ coNP would prove P ≠ NP
  - another major open problem…
coNP

• canonical coNP-complete language:
  \[\text{UNSAT} = \{\phi : \phi \text{ is an unsatisfiable } 3\text{-CNF formula}\}\]
  – proof?
coNP

- another example

3-DNF-TAUTOLOGY = \{\varphi : \varphi \text{ is a 3-DNF formula and for all } x, \varphi(x) = 1\}

- proof?

- another example:

EQUIV-CIRCUIT = \{(C_1, C_2) : C_1 \text{ and } C_2 \text{ are Boolean circuits and for all } x, C_1(x) = C_2(x)\}

- proof?