CS21
Decidability and Tractability
Lecture 23
February 28, 2022

Outline
• NP-complete problems: NAE-3-SAT, max cut
• The complexity class coNP
• The complexity class coNP \cap NP
• The complexity class PSPACE
  – a PSPACE-complete problem
  – PSPACE and 2-player games

Not-All-Equal 3SAT
\((x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_5) \land \ldots \land (\ldots)\)

**Theorem:** the following language is NP-complete:
NAE3SAT = \(\{ \phi : \phi \) 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 \(\in\) NP. Proof?
  – Part 2: NAE3SAT is NP-hard. Reduce from?

NAE3SAT is NP-complete
• Recall reduction to 3SAT
  – variables \(x_1, x_2, \ldots, x_n\) gates \(g_1, g_2, \ldots, g_m\)
  – produce clauses:
    \[\neg z_i \lor v \lor g_i\]
    \[\neg \neg z_i \lor v \lor \neg g_i\]
    \[\neg \neg g_i \lor z_1 \lor z_2\]
    \[\neg z_1 \lor \neg z_2 \lor v \lor g_i\]
    \[g_i\]
  – not all true in a satisfying assignment

NAE3SAT is NP-complete
• Recall reduction to 3SAT
  – variables \(x_1, x_2, \ldots, x_n\) gates \(g_1, g_2, \ldots, g_m\)
  – produce clauses:
    \[\neg z_i \lor v \lor g_i\]
    \[\neg \neg z_i \lor v \lor \neg g_i\]
    \[\neg \neg g_i \lor z_1 \lor z_2\]
    \[\neg z_1 \lor \neg z_2 \lor v \lor g_i\]
    \[g_i\]
NAE3SAT is NP-complete

• Does the reduction run in polynomial time?
  • NO maps to NO
    – given NAE assignment \(A\)
    – complement \(A'\) is a NAE assignment
    – \(A\) or \(A'\) has \(w = \text{FALSE}\)
    – must have TRUE \(\text{BLUE}\) variable in every clause
    – we know this implies \(C\) satisfiable

• YES maps to YES
  – already know how to get a satisfying assignment to the \(\text{BLUE}\) variables
  – set \(w = \text{FALSE}\)
  • \((\neg z_1 \lor g_i \lor w)\)
  • \((\neg z_2 \lor g_i \lor w)\)
  • \((\neg g_i \lor z_1 \lor z_2)\)
  • \((\neg g_i \lor z_1 \lor w)\)
  • \((\neg z_1 \lor \neg z_2 \lor g)\)
  • \((g_i \lor z \lor w)\)
  • \((\neg z \lor \neg g_i \lor w)\)
  • \((g_i \lor w)\)

MAX CUT

• Given graph \(G = (V, E)\)
  – a cut is a subset \(S \subseteq V\)
  – 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 is NP-complete

• We are reducing from the language:
  NAE3SAT = \(\{\phi : \phi \text{ is a 3-CNF formula for which there exists a truth assignment in which every clause has at least 1 true literal and at least 1 false literal}\}\)

  to the language:
  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):
    \((x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_4 \lor x_5) \land \ldots \land (\neg x_2 \lor x_3 \lor x_5)\)
  - produce graph \(G = (V, E)\) with node for each literal

- **triangle for each 3-clause**
- **parallel edges for each 2-clause**

**Claim:** 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 \(n\) parallel edges from \(x_i\) to \(\neg x_i\) (\(n = \#\) occurrences)
  - repeat variable in 2-clause to make 3-clause for this calculation

**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

**Claim:** if cut has \(x_i, \neg x_i\) on same side, then can move one to opposite side without decreasing # edges crossing cut

- set \(k = 5m\)

**NO maps to NO**
- contribution from \((x_i, \neg x_i)\) parallel edges: 3m
- contribution from clause gadgets must be 2m
- conclude: there is a NAE assignment

**coNP**

- Is NP closed under complement?
  - Can we transform this machine:
  - into a machine with this behavior?
coNP

- language L is in \textit{coNP} iff its complement (co-L) is in NP

- it is believed that \textit{NP} \neq \textit{coNP}
  - note: P = NP implies \textit{NP} = \textit{coNP}
    - proving \textit{NP} \neq \textit{coNP} would prove P \neq NP
    - another major open problem...

coNP

- canonical coNP-complete language:
  \textit{UNSAT} = \{ \varphi : \varphi \text{ is an unsatisfiable 3-CNF formula} \}
  - proof?

- another example:
  \textit{3-DNF-TAUTOLGY} = \{ \varphi : \varphi \text{ is a 3-DNF formula and for all x, } \varphi(x) =1 \}
  - proof?

- another example:
  \textit{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?

Quantifier characterization of coNP

- recall that a language L is in \textit{NP} if and only if it is expressible as:
  \[ L = \{ x : \exists y, |y| \leq |x|^k, (x, y) \in R \} \]
  where R is a language in P.

**Theorem:** language L is in \textit{coNP} if and only if it is expressible as:
\[ L = \{ x : \forall y, |y| \leq |x|^k, (x, y) \in R \} \]
where R is a language in P.