# COMPSCI 501: Formal Language Theory

## Lecture 24: NP-complete Problems

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

25 March 2019

## Reducing from *3SAT*

To show a problem $Q$ is NP-complete, we can show
1. $Q$ is in NP
2. *3SAT* $\leq_P Q$

3SAT is a good candidate: its structure is *simple* and *regular*

Many reductions use *gadgets*: fragments of the target problem that can represent variables and clauses.

We need to:

- force each clause to be satisfied
- force consistency (exactly one of $x_i$ and $\overline{x_i}$ true

## Karp's 21 NP-complete Problems (1972)

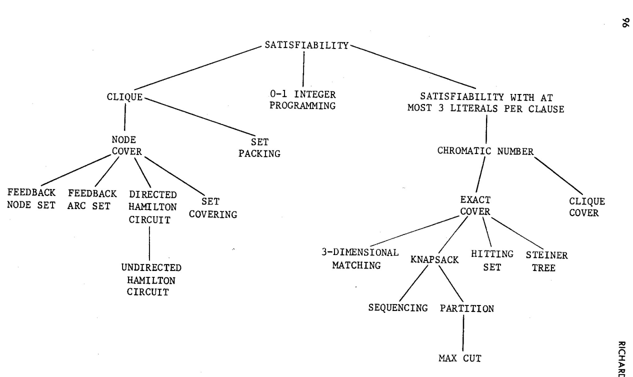


FIGURE 1 - Complete Problems

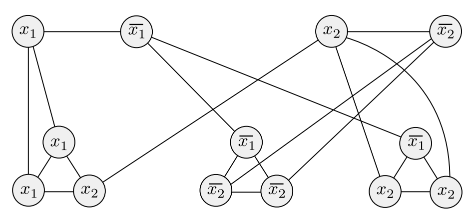
## Vertex Cover

A **vertex cover** of a graph $G$ is a set of nodes that contains an endpoint of every edge ("covers all edges").

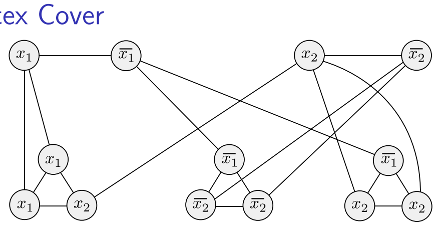
Given $G$ and $k$, does $G$ has a vertex cover of size $k$ ?

Intuition:

- one "triangles" for each clause: must choose $\geq 2$ nodes
  make one node dependent on satisfying formula
- one node per literal ($x_i$ and $\overline{x_i}$)
  connect to all literal occurrences



## Vertex Cover

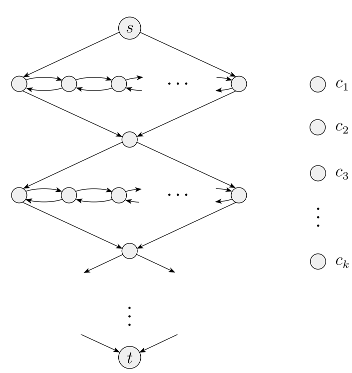


Assume $m$ variables and $c$ clauses. We choose $k = m + 2c$

If formula is satisfiable, mark true literals ($m$).
all their edges to clauses and complements are covered
Must still cover:
all clause edges
all edges from complements to clauses

In each clause, *don't* mark one literal in the cover,
but mark the other two (total: $2c$)
covers all triangles, and edges to unmarked variable nodes

## Hamiltonian Path

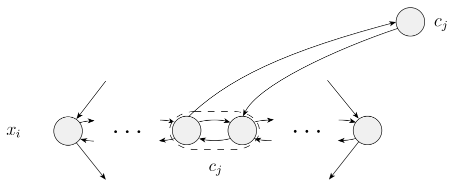


- one node per clause ($k$)
- one "diamond" per variable
- $3k + 1$ inner nodes per diamond (a pair per clause + one separator)

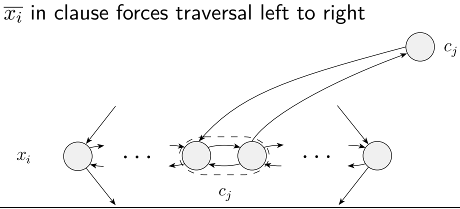
## Hamiltonian Path

Connect clause nodes to gadgets for each member variable

$x_i$ in clause forces traversal left to right




$\overline{x_i}$ in clause forces traversal left to right

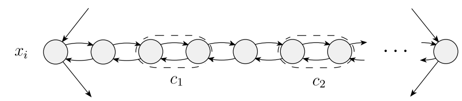



## Hamiltonian Path

Clause nodes can only be covered from their variable gadgets

A "diamond" can be traversed to the right or to the left, not both
variable is either true or false, only covers corresponding clauses

Can't return from clause node to different diamond,
because then the nodes between clause pairs won't be covered.




## Undirected Hamiltonian Path

Reduce directed *HAMPATH* $s \rightsquigarrow t$ to *UHAMPATH*.

Force direction in undirected graph by converting each node $u$ into connected triple $u^{\text{in}}$, $u^{\text{mid}}$, $u^{\text{out}}$.
Transform $s$ to $s^{\text{out}}$ and $t$ to $t^{\text{in}}$. Call new graph $G'$.

For each edge $u \to v$, introduce edge $u^{\text{out}}$ — $v^{\text{in}}$.

Clearly any directed $s \rightsquigarrow t$ Hamiltonian path passes through all triples.

Conversely, any *UHAMPATH* in $G'$ must start at $s^{\text{out}}$ and go to some $u^{\text{in}}$.
The next node must be $u^{\text{mid}}$, otherwise it will be skipped.
The only connection is then to $u^{\text{out}}$
We can repeat the argument by induction.

## Subset Sum

**Subset Sum**: Given a collection of integers $x_i$ and a target integer $t$, is there a subcollection that adds to $t$?

Reduction from 3-SAT. ($l$ variables, $k$ clauses, base 10).

- All numbers have $l + k$ digits
- Digits 1 to $l$: For variable $x_i$, create two items $t_i, f_i$
  - Both have $i$th digit equal to 1
  - All other numbers have this digit zero
  - $i$th digit of $t = 1 \Rightarrow$ must select exactly one of $t_i, f_i$
- The $l + j$th digit corresponds to clause $c_j$
  - If $x_i \in c_j$, set $l + j$th digit of $t_i = 1$
  - If $\neg x_i \in c_j$, set $l + j$th digit of $f_i = 1$
  - Everything else $0$.
- Choose $t$ with first $l$ digits 1 and last $k$ digits 3
- Create two "dummy" integers $g_j, h_j$ with $1$ in position $l + j$

## Subset Sum Example

**Example.**

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

| int | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $t_1$ | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| $f_1$ | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| $t_2$ | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| $f_2$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| $t_3$ | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| $f_3$ | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| $t$ | 1 | 1 | 1 | 3 | 3 | 3 | 3 |

| int | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $g_1$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $h_1$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $g_2$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $h_2$ | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| $g_3$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| $h_3$ | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| $g_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| $h_4$ | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

## Subset Sum Reduction

$\Rightarrow$

Consider a satisfying assignment.

Choose integer $t_i$ if $x_i$ true, and $f_i$ if $x_i$ false.
First $l$ columns add up.

In last $k$ columns, sum is between 1 and 3 (number of literals true per clause). Select 0 to 2 of the numbers $g_j, h_j$ to make the sum in column $l + j$ equal to 3.

$\Leftarrow$

Consider a collection adding up to $t$.

If must contain exactly one of $t_i$, $f_i$.

Since each of the last $k$ columns adds to 3, and at most two numbers $g_j$, $h_j$ were used, each column (clause) must have another 1 (satisfying assignment).