NP-Completeness

Independent Set

Input:
- Graph $G=(V,E)$ with n vertices, m edges
- integer $1 \leq k \leq n$

Solution: Independent set of size k (k vertices in V w/ no edges between them)

Algorithm: Try all sets of k vertices

$\#$ of sets $= \binom{n}{k} \approx n^k \Rightarrow$ runs in time $\Omega(n^k)$

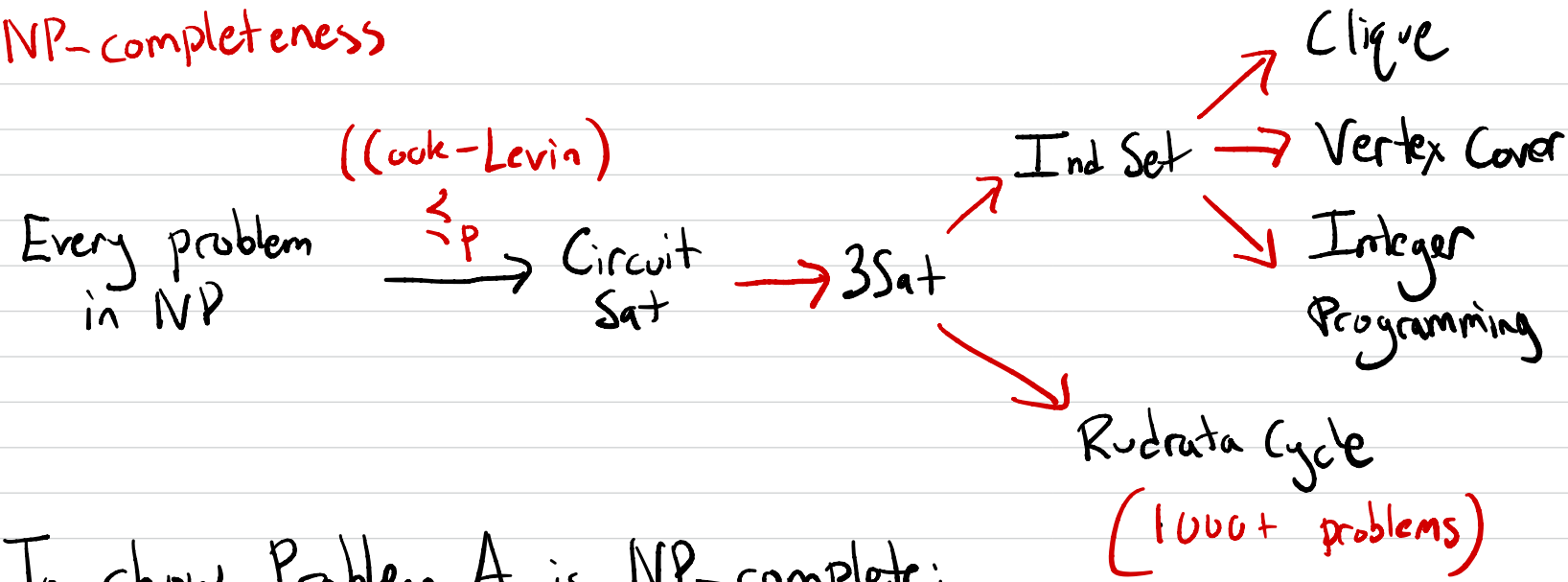
(good if $k = O(1)$, bad if $k = \omega(1)$)

Seems hard! How to show this?

Thm: Independent Set is NP-hard (+ NP-complete)

Pf: Pick some (well-known) NP-complete problem A and show that $A \leq_p$ Independent Set

NP-completeness



To show Problem A is NP-complete:

1.) $A \in NP$ (show verification algorithm)

2.) Show NP complete B reduces to A $(B \leq_p A)$

Shows poly-time alg for A $\Rightarrow$ poly-time alg for all of NP (unlikely!)

REDUCIBILITY AMONG COMBINATORIAL PROBLEMS†

Richard M. Karp

University of California at Berkeley

1972

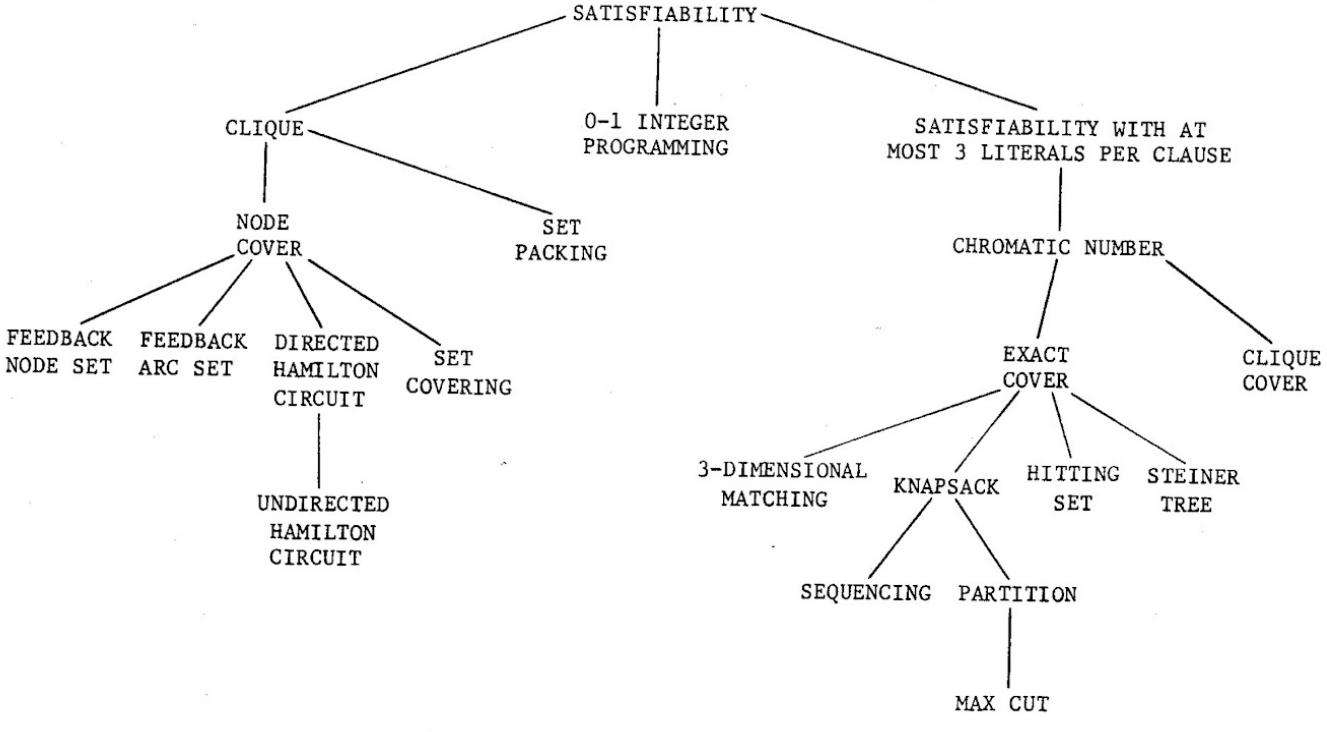


FIGURE 1 - Complete Problems

Thm: $3Sat \leq_p$ Ind-Set $(\therefore$ Ind-Set is NP-complete$)$

3Sat instance φ

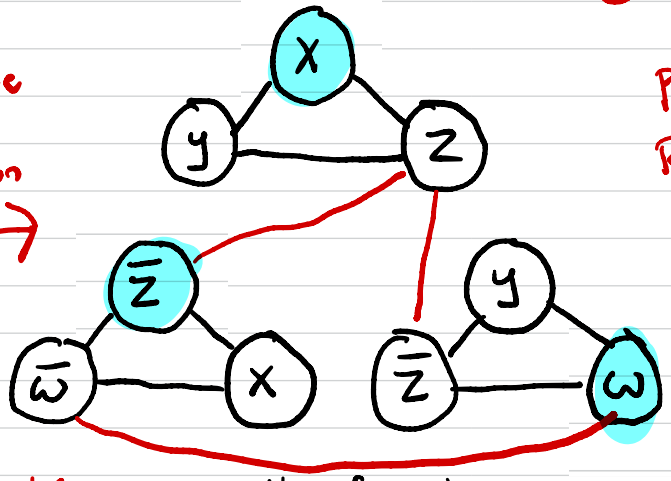
$$(x \vee y \vee z) \wedge (\bar{z} \vee \bar{w} \vee x) \wedge (y \vee \bar{z} \vee w)$$

(1 1 0 / 1 1 1 / 1 0 1)

Sat assign: $x=y=1$, $z=w=0$

poly-time Reduction →

Ind Set instance G



$K = m =$ # of clauses

poly-time Recovery →

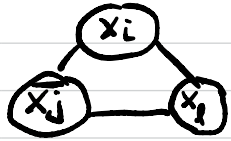
$x = 1$

$y = 0$ or 1

$z = 0$

$w = 1$

Proof part 1: If 3Sat inst x has sat assign $A: \{x_1, \ldots, x_n\} \to \{0,1\}$, then graph G has ind set I of size $k = m$

- For each clause $x_i \vee \bar{x_j} \vee x_l$, A sets either $x_i = 1, \bar{x}_j = 1, x_l = 1$.
- Pick one (say x_i). Add (x_i) of 

to I.
- So I is of size $k = m$.
- I contains no (x_i)—$(\bar{x_i})$. ∴ No edges, so independent set.

Proof part 2: Let I be ind set in G of size K. Then Recovery alg outputs satisfying assignment.

Recovery alg

$$\text{variable } x_i = \begin{cases} 1 & \text{if some } (x_i) \in I \\ 0 & \text{if some } (\bar{x_i}) \in I \\ \text{arbitrary} & \text{o.w.} \end{cases}$$

- For each i, only (x_i) or $(\bar{x_i}) \in I$. So Recovery alg well-defined.
- For each 

$x_i = 1, x_j = 1$, or $x_l = 1$.
So all clauses satisfied.

Integer Programming (IP)

Input: A linear program

Solution: An integer solution to LP

Thm: Ind-Set $\leq_p$ Integer Programming

Pf: Ind Set instance
- $G = (V, E)$
- $V = \{1, 2, \ldots, n\}$
- integer k

poly-time Reduction →

IP instance

$0 \leq x_i \leq 1$

$\sum_{i=1}^{n} x_i = k$

$\forall (i,j) \in E,\ x_i + x_j \leq 1$

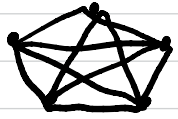
Proof part 1: If $\exists$ ind set I of size k, $\exists$ solution to IP instance. (Just set $x_i = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{if } i \notin I \end{cases}$)

Proof part 2: If $(x_1, \ldots, x_n)$ is solution to IP, can "recover" and ind set of size k. (Just put $i \in I$ if $x_i = 1$)

Clique

Input: • graph $G=(V,E)$
• integer $1 \le k \le n$

Output: Clique of size k → (k vertices in V in which every pair is adjacent)

 = 5-clique

Thm: Independent Set $\le_p$ Clique

Pf:

Ind set instance
• $G=(V,E)$
• integer k

Reduction alg →

Clique instance
• $\overline{G}=(V,\overline{E})$
$\overline{E}$ = set of edges not in E
• same integer k

Ind set in G $\therefore$ ⟷ Clique in $\overline{G}$ △

Recovery alg: Given clique $S \subseteq V$, output S.

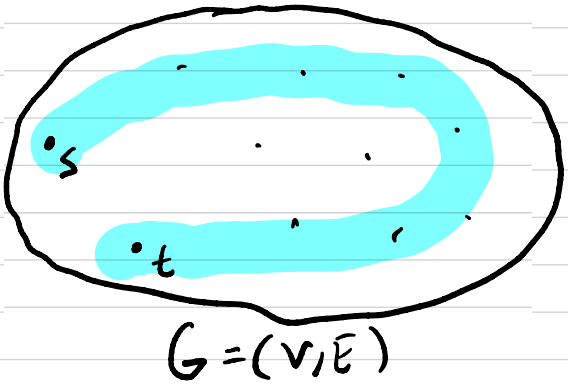
Rudrata (s,t) path

Input: Graph $G=(V,E)$, source $s \in V$, destination $t \in V$

Output: path starting at s, ending at t, visiting each vertex exactly once

Thm: Rudrata (s,t) path $\leq_P$ Rudrata Cycle

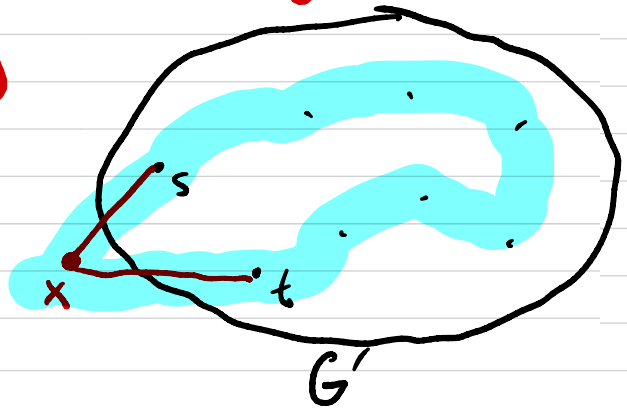
Pf: Rud (s,t) path instance



$G=(V,E)$

Reduction alg →

Rud cycle instance



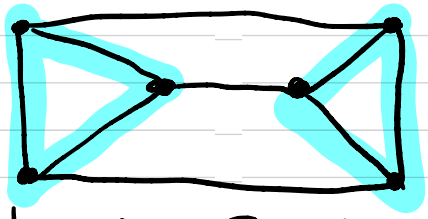
G'

Recovery alg: given cycle in G', delete (x,s) and (x,t) edges

Double Cycle

Input: Graph $G=(V,E)$ with $n=|V|=$ even

Output: two disjoint cycles of $n/2$ vertices each, which visit each vertex once



Thm: Rudrata Cycle $\leq_p$ Double Cycle

Pf: Rudrata instance → Reduction alg → Double Cycle Instance