Proving NP-Complete Problems

- The Cook-Levin Theorem: SAT is NP-Complete
- NP-Complete Problems from known Reductions
- DIR-HAM-CYCLE is NP-Complete
- DIR-HAM-PATH is NP-Complete
- HAM-CYCLE is NP-Complete
- TSP is NP-Complete
- SUBSET-SUM is NP-Complete
- PARTITION is NP-Complete

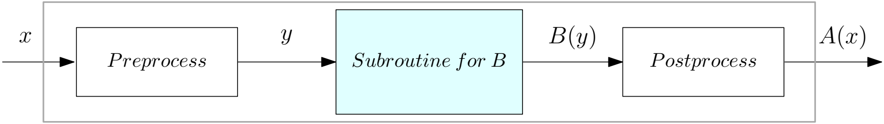
Imdad ullah Khan

Polynomial Time Reduction: Algorithm Design Paradigm

Problem A is polynomial time reducible to Problem B, $A \leq_p B$

If any instance of problem A can be solved using a polynomial amount of computation plus a polynomial number of calls to a solution of problem B

Subroutine for B takes an instance y of B and returns the solution $B(y)$



Algorithm for A transforms an instance x of A to an instance y of B. Then transforms $B(y)$ to $A(x)$

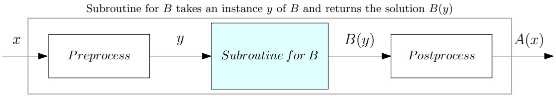
Suppose $A \leq_p B$.

If B is polynomial time solvable, then A can be solved in polynomial time

Reduction as a tool for hardness

Problem A is polynomial time reducible to Problem B, $A \leq_p B$

If any instance of problem A can be solved using a polynomial amount of computation plus a polynomial number of calls to a solution of problem B



Algorithm for A transforms an instance x of A to an instance y of B. Then transforms $B(y)$ to $A(x)$

A problem X is NP-COMPLETE, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

Suppose $A \leq_p B$.

If A is NP-COMPLETE, then B is NP-COMPLETE

▷ **Why?** By transitivity of reduction

Proving NP-Complete Problems

A problem X is NP-Complete, if

1. $X \in \text{NP}$
2. $\forall\ Y \in \text{NP}\ Y \leq_p X$

To prove X NP-Complete, reduce an NP-Complete problem Z to X

If Z is NP-Complete, and

1. $X \in \text{NP}$
2. $Z \leq_p X$

then X is NP-Complete

1. $X \in \text{NP}$ is explicitly proved
2. $\forall\ Y \in \text{NP},\quad Y \leq_p X$ follows by transitivity

   $\forall\ Y \in \text{NP},\quad Y \leq_p Z$ is true as Z is NP-Complete

   $[Y \leq_p Z \ \wedge\ Z \leq_p X] \implies Y \leq_p X$

Proving NP-Complete Problems

A problem X is NP-Complete, if

1. $X \in \mathrm{NP}$
2. $\forall\ Y \in \mathrm{NP}\ Y \leq_p X$

How to prove a problem NP-Complete?

To prove X to be NP-Complete

1. Prove $X \in \mathrm{NP}$
2. Reduce some known NP-Complete problem Z to X

Again! Reduce a known NP-Complete problem to X

▷ **Not the other way round. A very common mistake!**

A first NP-Complete Problem

Theorem (The Cook-Levin theorem)

SAT(f) is NP-Complete

- Proved by Stephen Cook (1971) and earlier by Leonid Levin (but became known later)
- Levin proved six NP-Complete problems (in addition to other results)
- We prove this by reducing CIRCUIT-SAT(C) problem to SAT(f) problem

A first NP-Complete Problem

To prove X NP-Complete, reduce an NP-Complete problem Z to X

Where to begin? we need a first NP-Complete Problem

Theorem (The Cook-Levin theorem)

SAT(f) *is* NP-Complete

- Proved by Stephen Cook (1971) and earlier by Leonid Levin (but became known later)
- Levin proved six NP-Complete problems (in addition to other results)

We prove the theorem by reducing CIRCUIT-SAT(C) problem to SAT(f) problem

The Cook-Levin theorem

Theorem (The Cook-Levin theorem)

SAT(f) *is* NP-COMPLETE

We already showed that SAT is polynomial time verifiable

SAT $\in$ NP

Now we prove that

CIRCUIT-SAT(C) $\leq_p$ SAT(f)

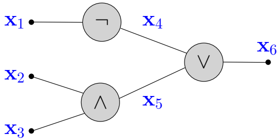
This proves that SAT is NP-HARD and completes the proof

- Suppose $\mathcal{A}$ is an algorithm to decide SAT(f)
- Given an instance C of the CIRCUIT-SAT(C) problem
- In polynomial time we transform C into an equivalent CNF formula f
- Make a call $\mathcal{A}(f)$ to decide whether or not CIRCUIT-SAT(C) = **Yes**

The Cook-Levin theorem

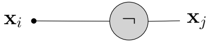
$$\text{CIRCUIT-SAT}(C) \leq_p \text{SAT}(f)$$

Make a variable for each input wire and output of each gate of the circuit C



For each not gate make equi-satisfiable clauses

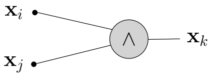
- These clauses are satisfied iff $x_j = \overline{x_i}$



$$(x_i \vee x_j) \wedge (\overline{x_i} \vee \overline{x_j})$$

For each and gate make equi-satisfiable clauses

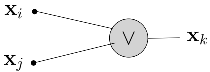
- These clauses are satisfied iff $x_k = x_i \wedge x_j$



$$(x_i \vee \overline{x_k}) \wedge (x_j \vee \overline{x_k}) \wedge (\overline{x_i} \vee \overline{x_j} \vee x_k)$$

For each or gate make equi-satisfiable clauses

- These clauses are satisfied iff $x_k = x_i \vee x_j$



$$(\overline{x_i} \vee x_k) \wedge (\overline{x_j} \vee x_k) \wedge (x_i \vee x_j \vee \overline{x_k})$$

The Cook-Levin theorem

$\textsc{circuit-sat}(C) \quad \leq_p \quad \textsc{sat}(f)$

- Easy to verify that the gates and corresponding formula are equisatisfiable
- The output gate value is encoded with a clause containing the corresponding variable
- The final formula f is a grand conjunction of all the clauses made for each gate and output of the circuit C

f is equisatisfiable with the C

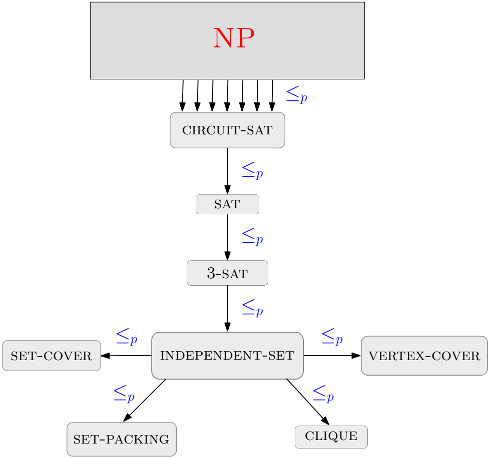
▷ i.e. $\textsc{circuit-sat}(C) =$ **Yes** if and only if $\mathcal{A}(f) =$ **Yes**

The reduction takes polynomial time, requires one traversal of the DAG, constant time per gate

Implied NP-COMPLETE Problems

From known reductions, the following problems are NP-COMPLETE

- SAT $\leq_p$ 3-SAT
- 3-SAT $\leq_p$ IND-SET
- IND-SET $\leq_p$ CLIQUE
- IND-SET $\leq_p$ VERTEX-COVER
- VERTEX-COVER $\leq_p$ SET-COVER
- IND-SET $\leq_p$ SET-PACKING



We show a few more reductions to prove problems to be NP-COMPLETE

DIR-HAM-CYCLE is NP-COMPLETE

We showed DIR-HAM-CYCLE to be in NP for NP-HARDNESS we prove

$$3\text{-SAT}(f) \leq_p \text{DIR-HAM-CYCLE}(G)$$

Let f be an instance of 3-SAT on n variables and m clauses

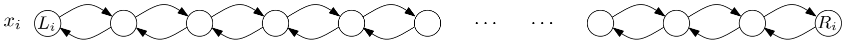
Let $x_1, \ldots, x_n$ be the variables and $C_1, \ldots, C_m$ be the clauses of f

Construct a digraph G that has a Hamiltonian cycle iff f is satisfiable

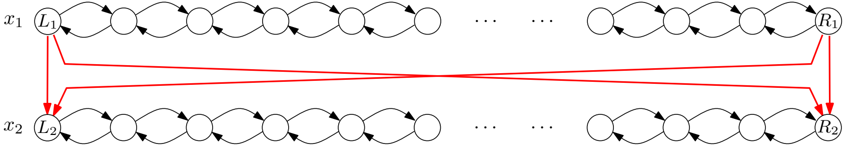
1. In G there will be 2^n sub-Hamiltonian cycles corresponding to the 2^n possible assignments to variables $x_1, \ldots, x_n$
2. We introduce a structure for each clause such that these sub-Hamiltonian cycles can be combined if and only if all clauses are satisfiable

3-SAT(f) $\leq_p$ DIR-HAM-CYCLE(G)

For each x_i make a sequence of $3(m+1)$ bidirectionally adjacent vertices



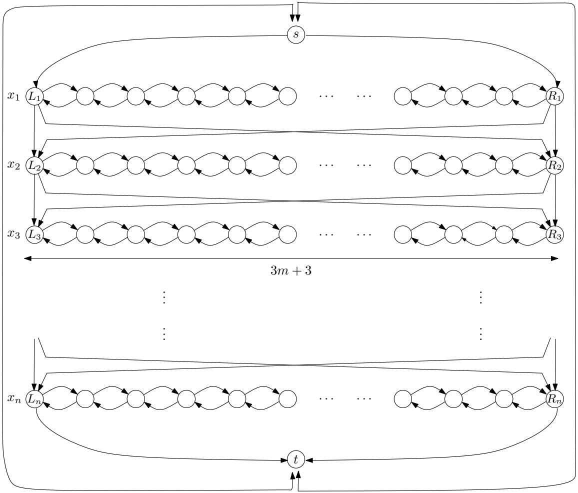
- $x_i = 1 \implies$ traverse this gadget from L_i to R_i and vice-versa
- $(x_i, x_{i+1}) = (1, 0) \implies$ traverse from $L_i \to R_i \to R_{i+1} \to L_{i+1}$
- $(x_i, x_{i+1}) = (0, 0) \implies$ traverse from $R_i \to L_i \to R_{i+1} \to L_{i+1}$



DIR-HAM-CYCLE is NP-COMPLETE

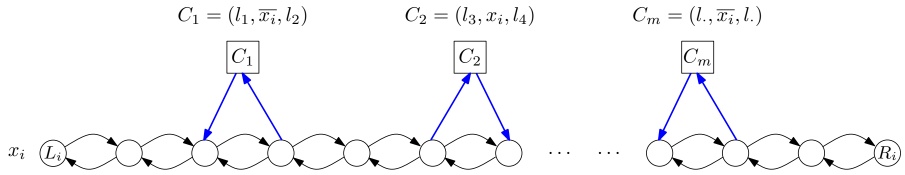
$$3\text{-SAT}(f) \quad \leq_p \quad \text{DIR-HAM-CYCLE}(G)$$

Make nodes s and t and combine all the gadgets as follows



3-SAT(f) $\leq_p$ DIR-HAM-CYCLE(G)

- 2^n Ham cycles traversing each gadget in either direction
- These correspond to the 2^n possible assignments to the n variables
- Make a Hamiltonian cycle exist iff there is a satisfying assignment
- Have to incorporate clauses. Make nodes for each clause
- If a variable satisfy a clause, traverse it by a detour from that gadget



DIR-HAM-CYCLE is NP-COMPLETE

3-SAT(f) $\leq_p$ DIR-HAM-CYCLE(G)

Given f, make G as described above

G has a directed Hamiltonian cycle iff f is satisfiable

The construction takes polynomial time (about $O(nm)$)

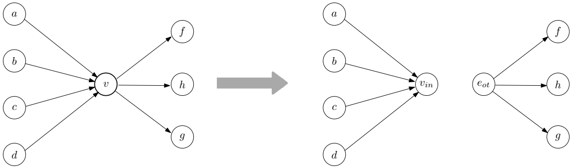
DIR-HAM-CYCLE(G) $\leq_p$ DIR-HAM-PATH(G')

Let $G = (V, E)$ be an instance of the DIR-HAM-CYCLE(G) problem

- For any arbitrary $v \in V$, make G' on $V(G) \setminus \{v\} \cup \{v_{in}, v_{out}\}$

   ▷ i.e. remove v and add two new vertices v_{in} and v_{out}

- v_{in} has all incoming edges of v directed to it from in-neighbors of v
- v_{out} has all outgoing edges of v directed from it to out-neighbors of v



G has a directed Hamiltonian cycle iff G' has a directed Hamiltonian path

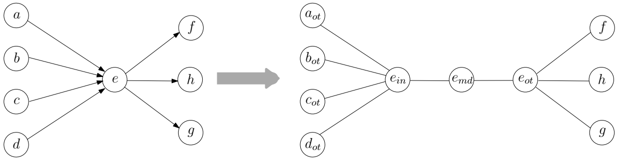
HAM-CYCLE is NP-COMPLETE

We proved its polynomial time verifiability earlier, now we show that

DIR-HAM-CYCLE(G) $\leq_p$ HAM-CYCLE(G')

Let $G = (V, E)$ be an instance of the DIR-HAM-CYCLE(G). $|V| = n$, $|E| = m$

- Make an undirected graph $G' = (V', E)$, $|V'| = 3n$ and $|E'| = m + 2n$
- Split every vertex $v \in V$ into three vertices v_{in}, v_{md}, v_{ot} and add to V'
- Add edges (v_{in}, v_{md}) and (v_{md}, v_{ot}) in E'
- For each directed edge $(x, y) \in E$, make the edge (x_{ot}, y_{in}) in E'



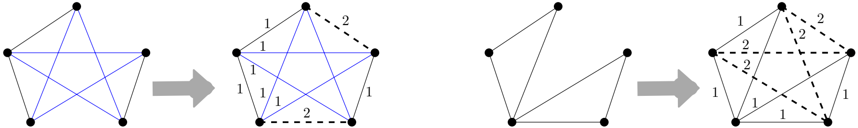
G has a dir-Ham cycle iff G' has an (undirected) Hamiltonian cycle

TSP is NP-Complete

$$\text{HAM-CYCLE}(G) \leq_p \text{TSP}(G', k)$$

- TSP(G', k) requires weighted graph and a number k
- Given an instance $G = (V, E)$ of HAM-CYCLE(G), $|V| = n$
- Make a complete graph on n vertices G' with weights as follows

$$w(v_i, v_j) = \begin{cases} 1 & \text{if } (v_i, v_j) \in E(G) \\ 2 & \text{else} \end{cases}$$



Hamiltonian cycle in G shown in blue

TSP tour in G' of of length shown in blue

No Hamiltonian cycle in G

No TSP tour of length 5 in G'

G has a Hamiltonian cycle iff G' has a TSP tour of length $k = n$

SUBSET-SUM is NP-COMPLETE

SUBSET-SUM is NP-COMPLETE

- Given a set $U = \{a_1, a_2, \dots, a_n\}$ of integers
- A weight function $w : U \to \mathbb{Z}^+$, and a positive integer C
- The SUBSET-SUM(U, w, C) problem: Is there a $S \subset U$ wiht $\sum_{a_i \in S} w_i = C$?
- If w_i's and C are given in unary encoding
  - then $O(nC)$ dynamic programming solution is a polynomial time
- But this is exponential in size of input if C is provided in binary (or decimal)

We prove that

$$3\text{-SAT}(f) \quad \leq_p \quad \text{SUBSET-SUM}(\bullet, \bullet, \bullet)$$

$3\text{-SAT}(f) \;\; \leq_p \;\; \text{SUBSET-SUM}(\bullet, \bullet, \bullet)$

- Given an instance f of $3\text{-SAT}(f)$ with n variables and m clauses
- Construct $2n + 2m$ weights: 2 objects for each variable and each clause
- Each is a $n + m$-digits integer (a digit for each variable and each clause)
- The weight for literal x_i and $\overline{x_i}$ have digit 1 corresponding to the variable x_i
- The digit for clause C_j is 1 if the literal appears in clause C_j

	x_1	x_2	x_3						x_{n-1}	x_n	C_1	C_2	C_3		C_{m-1}	C_m
x_1	1							$\cdots$			1	1		$\cdots$		
$\overline{x_1}$	1							$\cdots$					1	$\cdots$		
x_2		1						$\cdots$			1	1		$\cdots$	1	1
$\overline{x_2}$		1						$\cdots$					1	$\cdots$		
x_3			1					$\cdots$						$\cdots$		1
$\overline{x_3}$			1					$\cdots$			1	1	1	$\cdots$		
$\vdots$				$\vdots$							$\vdots$				$\vdots$	

$C_1 = (x_1 \vee x_2 \vee \overline{x_3}),\; C_2 = (x_1 \vee x_2 \vee \overline{x_3}),\; C_3 = (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$

$C_{m-1} = (x_2 \vee x_8 \vee x_9),\; C_m = (x_2 \vee x_3 \vee x_5)$

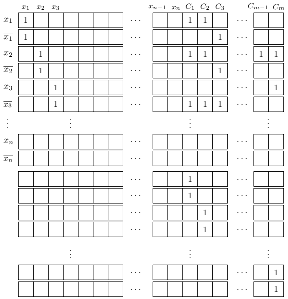
$$3\text{-SAT}(f) \quad \leq_p \quad \text{SUBSET-SUM}(\bullet, \bullet, \bullet)$$

- Remaining $2m$ weights set so as last sum of digits at each position from $n+1$ to $n+m$ is 5 ▷ details in notes

	x_1	x_2	x_3						x_{n-1}	x_n	C_1	C_2	C_3		C_{m-1}	C_m
x_1	1							$\cdots$			1	1		$\cdots$		
$\overline{x_1}$	1							$\cdots$					1	$\cdots$		
x_2		1						$\cdots$			1	1		$\cdots$	1	1
$\overline{x_2}$		1						$\cdots$					1	$\cdots$		
x_3			1					$\cdots$						$\cdots$		1
$\overline{x_3}$			1					$\cdots$			1	1	1	$\cdots$		
$\vdots$			$\vdots$								$\vdots$				$\vdots$	
x_n								$\cdots$						$\cdots$		
$\overline{x_n}$								$\cdots$						$\cdots$		
								$\cdots$			1			$\cdots$		
								$\cdots$			1			$\cdots$		
								$\cdots$				1		$\cdots$		
								$\cdots$				1		$\cdots$		
			$\vdots$								$\vdots$				$\vdots$	
								$\cdots$						$\cdots$		1
								$\cdots$						$\cdots$		1

SUBSET-SUM is NP-COMPLETE

3-SAT(f) $\leq_p$ SUBSET-SUM$(\bullet, \bullet, \bullet)$



The SUBSET-SUM instance with $2n + 2m$ weights as shown above and

$C = \overbrace{111\ldots,11}^{n}\overbrace{333\ldots33}^{m}$ is **Yes** if and only the f is satisfiable

SUBSET-SUM(U, w, C) $\leq_p$ PARTITION(U', k)

- Let $U' = \{w_1, w_2, \ldots, w_n\ ,\ \ w_{n+1}\ ,\ \ w_{n+2}\}$
- $w_{n+1} = 2\left[\sum_{i=1}^{n} w_i\right] - C$ and $w_{n+2} = \left[\sum_{i=1}^{n} w_i\right] + C$

SUBSET-SUM(U, w, C) = **Yes** iff PARTITION$(U', \mathbf{0})$ = **Yes** (balanced)

- $$\sum_{x \in U'} x = \sum_{a_i \in U} w_i + \underbrace{2\left[\sum_{i=1}^{n} w_i\right] - C}_{w_{n+1}} + \underbrace{\left[\sum_{i=1}^{n} w_i\right] + C}_{w_{n+2}} = 4 \sum_{a_i \in U} w_i$$
- Let P_1 and P_2 be a balanced bipartition of U'
- Both w_{n+1} and w_{n+2} cannot be in the same part, assume $w_{n+1} \in P_1$
- Both P_1 and P_2 cannot contain only one element, so $\sum_{x \in P_1 \setminus \{w_{n+1}\}} w_x = C$

P_1		P_2	
$w_{n+1} = 2\sum_i w_i - C$	C	$w_{n+2} = \sum_i w_i + C$	$\sum_i w_i - C$

NP-COMPLETE Problems

21 problems were shown to be NP-COMPLETE in a seminal paper: Richard Karp (1972), "Reducibility Among Combinatorial Problems"

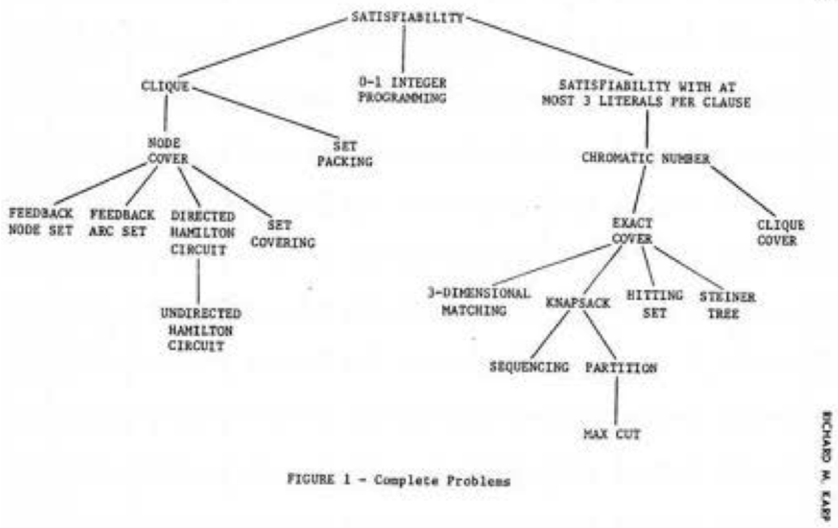


FIGURE 1 - Complete Problems