

NP Hardness - 2

Outline

Clique
Graph coloring
Hamiltonian cycle

Described a model of computation, known today as the Turing Machine.

A problem P is *decidable* (or *computable*) if it can be solved by a Turing machine that halts on every input.

We say that P has an algorithm.

There are undecidable problems - the Halting problem - for which it is known to be impossible to construct an algorithm that always leads to a yes-or-no answer.



Alan Turing
(1936, age 22)

The Church-Turing Thesis

"Whatever is computable is computable by a Turing machine."

This is not a theorem.

Everyone believes it.

No counterexample yet.

The discussion of what constitutes computability is closed: all the models are equivalent.

Complexity Classes



A fundamental complexity class P (or $PTIME$) is a class of decision problems that can be solved by a deterministic Turing machine in polynomial time.

Complexity Class: NP

A fundamental complexity class NP is a class of decision problems that can be solved by a nondeterministic Turing machine in polynomial time.



Equivalently, the NP decision problem has a *certificate* that can be checked in a polynomial time by deterministic Turing machine.

NP -problems like FIND a needle in a haystack: hard to find but always easy to verify

P versus NP

It has been proven that Nondeterministic TM can be simulated by Deterministic TM.

But how fast we can do that simulation?

The famous $P \neq NP$ conjecture, would answer that we cannot hope to simulate nondeterministic Turing machines very fast (in polynomial time).

Polynomial Reduction: $Y \leq_p X$

To reduce a decision problem Y to a decision problem X (we write $Y \leq_p X$) we want a function f that maps Y to X such that:

- 1) f is a polynomial time computable
- 2) $\forall y \in Y$ (y is instance of Y) is YES if and only if $f(y) \in X$ is YES.

Two ways of using $Y \leq_p X$

1) X is easy

If we can solve X in polynomial time, we can solve Y in polynomial time.

2) Y is hard

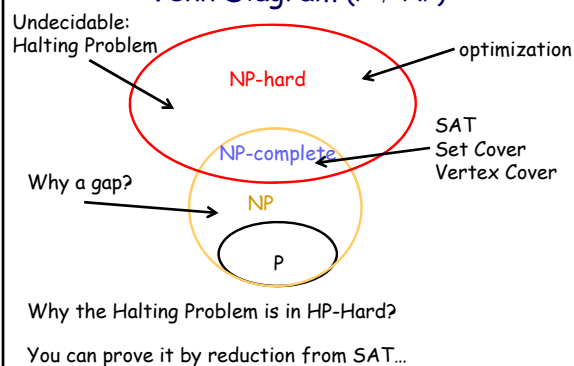
Then X is at least as hard as Y

NP-Hard and NP-Complete

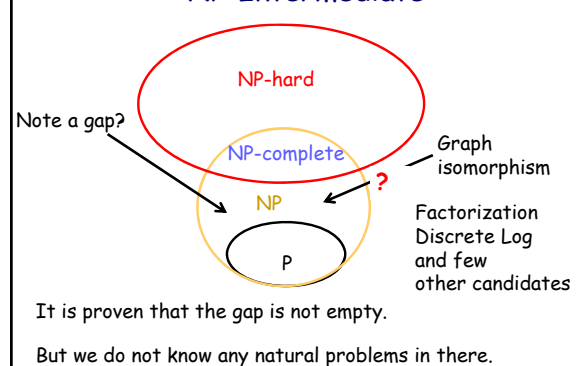
X is *NP-Hard*, if $\forall Y \in NP$ and $Y \leq_p X$.

X is *NP-Complete*, if X is NP-Hard and $X \in NP$.

Venn Diagram ($P \neq NP$)

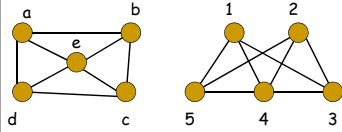


NP-Intermediate



Graph Isomorphism

Two graphs $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ are **isomorphic** if there is a bijective function $f: V_1 \rightarrow V_2$ such that $(v,w) \in E_1 \Leftrightarrow \{f(v),f(w)\} \in E_2$

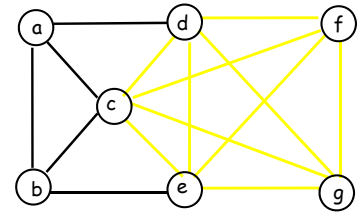


The two graphs look differently but are structurally the 'same', up to the renaming of the vertices.

Problem is in NP, but

- No NP-completeness proof is known
- No polynomial time algorithm is known

CLIQUE



k -clique = complete subgraph of k nodes

Given a graph and a number k , does the graph contain a k -clique?

$3SAT \leq_p k\text{-CLIQUE}$

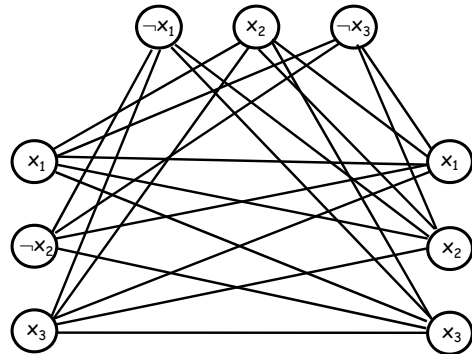
Construction:

A node for each literal.

An edge between literals that can be simultaneously true and in different clauses.
A k -clique will be a set of literals, one per clause, that can all be true simultaneously.

Claim: 3SAT is satisfiable if G has a k -clique.

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



How do we find a 3-clique?

$3SAT \leq_p k\text{-CLIQUE}$

Claim: 3SAT is satisfiable if G has a k -clique.

Proof.

\Rightarrow) If 3SAT is satisfiable, then there is a setting of the variables with at least one literal per clause set to true. Let Z be such a set of literals.

This set Z cannot contain both x and $\neg x$, so in the graph G , the nodes in Z have an edge between each pair and therefore form a k -clique.

$3SAT \leq_p k\text{-CLIQUE}$

Claim: 3SAT is satisfiable if G has a k -clique.

Proof.

\Leftarrow) If G has a k -clique, the clique must consist of k nodes and must not have x and $\neg x$

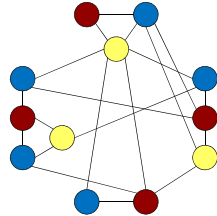
Therefore we can set the corresponding literals in a clique to true and satisfy 3SAT.



Graph Coloring

Given a graph, can you color the nodes with $\leq k$ colors such that the endpoints of every edge are colored differently?

Theorem. ($k \geq 2$)
k-Coloring is NP-complete.



Graph Coloring: $k = 2$

How can we test if a graph has a 2-coloring?

We can do this by checking if the graph is bipartite.

Alternatively, color G in the level order traversal.

3-SAT \leq_p 3-colorable

We construct a graph G that will be 3-colorable iff the 3-SAT instance is satisfiable.

Graph G consists of the following gadgets.

A truth gadget:

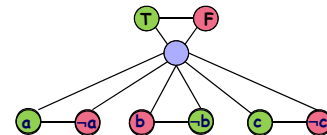


A gadget for each variable:



3-SAT \leq_p 3-colorable

Combining those gadgets together:



Any 3-coloring of the above subgraph defines a valid truth assignment! And vice versa.

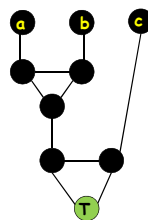
We have to make sure that the truth assignments satisfy the given clauses.

3-SAT \leq_p 3-colorable

A special gadget for each clause

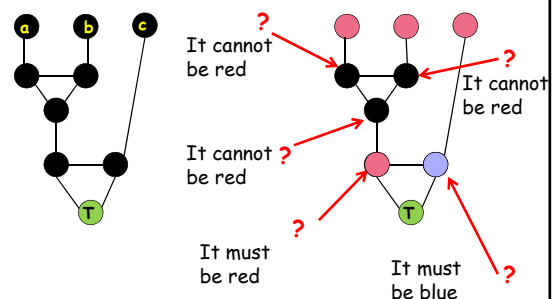
The bottom node is colorable as T (green) iff one of the inputs (a, b or c) is the same color as T.

Let us prove this.



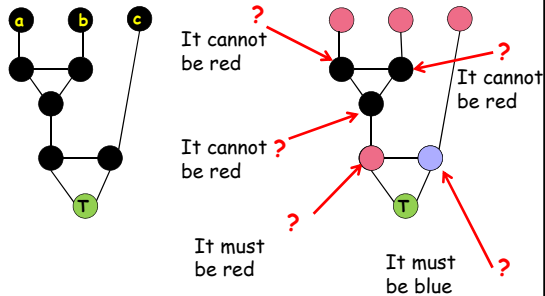
3-SAT \leq_p 3-colorable

Suppose all a, b and c are all False (red).

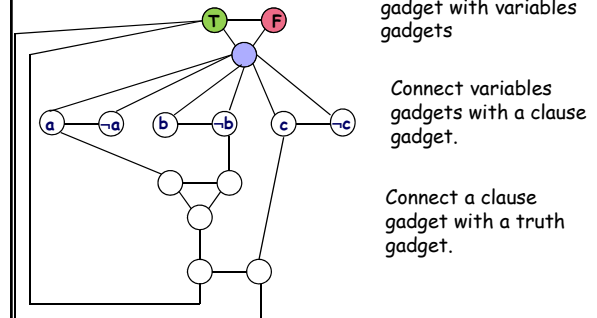


3-SAT \leq_p 3-colorable

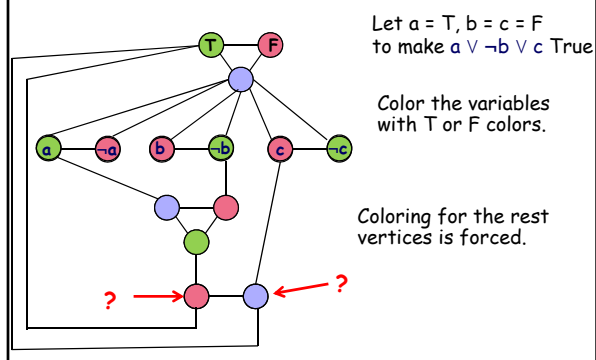
We have showed that if all the variables in a clause are false, the gadget cannot be 3-colored



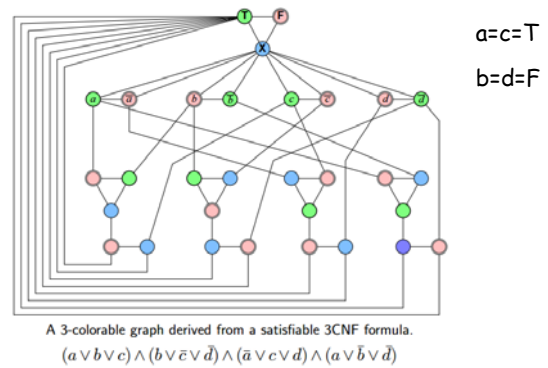
Example: $a \vee \neg b \vee c$



Example: $a \vee \neg b \vee c$



Example with four clauses



3-SAT \leq_p 3-colorable

Claim: 3-SAT instance is satisfiable if and only if G is 3-colorable.

Proof: \rightarrow)

Given a satisfying assignment for 3-SAT.

We color the truth gadget with T, F and blue.

We color the variables with T or F according to the assignment.

Coloring for the rest vertices is forced.


3-SAT \leq_p 3-colorable

Claim: 3-SAT instance is satisfiable if and only if G is 3-colorable.

Proof: \leftarrow)

Given a 3-coloring for that graph.

Choose green for T, red for F.



Sudoku: $n^2 \times n^2$

NP-?

NP-hard?

Sudoku graph?

2			3	8		5		
		3		4	5	9	8	
		8			9	7	3	4
6		7		9				
9	8						1	7
				5	6			9
3	1	9	7			2		
	4	6	5	2		8		
	2		9	3				1

Sudoku graph: vertex is each cell (81 of them), two vertices connected by an edge, if they are in the same row, column and small grid.

Sudoku Graph

What is the degree of a vertex in the Sudoku graph?

$20 = 8 + 8 + 8$

How many edges in the Sudoku graph?

$810 = 81 \cdot 20 / 2$

2			3	8		5		
		3		4	5	9	8	
		8			9	7	3	4
6		7		9				
9	8						1	7
				5	6			9
3	1	9	7			2		
	4	6	5	2		8		
	2		9	3				1

Sudoku

Constructing a Sudoku graph we have proved:

Sudoku \leq_p 9-colorable

Did we prove that Sudoku is NP-complete?

No, we did not!

How this reduction is very useful!

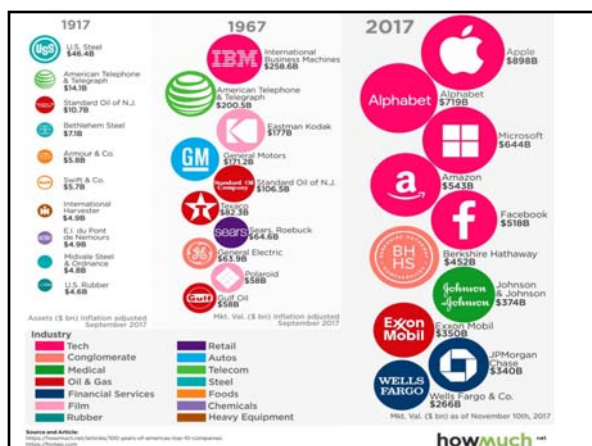
It is useful for SOLVING the Sudoku game!!!!

Best Global Universities for Computer Science



#18 in the world!!!


- #1 Tsinghua University
 - China Beijing
 - #64 – Best Global Universities
- #18 University of Southern California
 - United States Los Angeles, CA
 - #62 – Best Global Universities
- #19 Princeton University
 - United States Princeton, NJ
 - #9 – Best Global Universities



Hamiltonian Cycle Problem

A Hamiltonian cycle (HC) in a graph is a cycle that visits each vertex exactly once.

Problem Statement:
 Given a *directed* graph $G = (V, E)$
 Find if the graph contains a Hamiltonian cycle.



A Hamiltonian cycle problem is NP-Complete

A Hamiltonian cycle problem is in NP. Easy!

A Hamiltonian cycle problem is in NP-Hard.

We will prove it by reduction 3-SAT \leq_p HC.

Given a 3-CNF formula Φ , we want to construct a directed graph from Φ with the following properties:

1. a satisfying assignment to Φ translates into a Hamiltonian cycle
2. a Hamiltonian cycle can be translated into a satisfying assignment

3-SAT \leq_p HC

We begin with an arbitrary instance of 3-SAT having variables X_1, \dots, X_n and clauses C_1, \dots, C_m

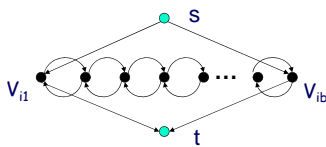
$$(X_1 \vee \neg X_3 \vee \neg X_4) \wedge (X_1 \vee \neg X_2 \vee X_4) \wedge \dots$$

Since there are 2^n assignments, we create a graph containing 2^n different Hamiltonian cycles.

We will build the graph up from pieces called **gadgets** that "simulate" the clauses and variables.

The variable gadget (one for each X_i)

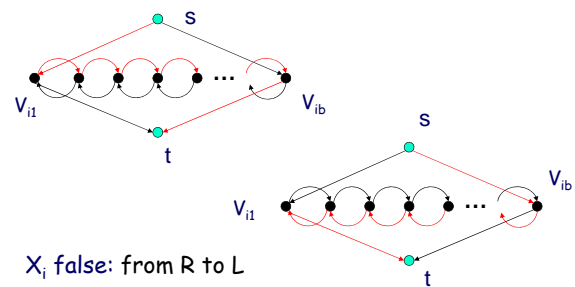
For each variable X_i we create a gadget (a cross-bar) with $b = 2m$ (m is # of clauses) vertices $V_{i1}, V_{i2}, \dots, V_{ib}$ and with edges going in both directions.



We also added two special vertices (at the top and bottom)

The variable gadget (one for each X_i)

X_i true: we traverse the gadget from Left to Right



Example

$$(X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_2 \vee X_3 \vee X_4) \wedge (\neg X_1 \vee X_2 \vee \neg X_4)$$

$$n = 4, k = 3, b = 2 \cdot 3 = 6$$

$$b = 2m \text{ (} m \text{ is \# of clauses)}$$

Construct 4 gadgets:

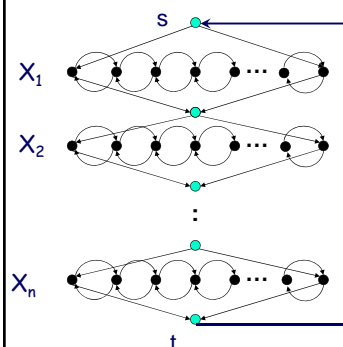
X_1 consists of nodes $V_{1,1}, V_{1,2}, \dots, V_{1,6}$

X_2 consists of nodes $V_{2,1}, V_{2,2}, \dots, V_{2,6}$

X_3 consists of nodes $V_{3,1}, V_{3,2}, \dots, V_{3,6}$

X_4 consists of nodes $V_{4,1}, V_{4,2}, \dots, V_{4,6}$

The gadget for all variables



A path from s to t translates into a truth assignment to X_1, \dots, X_n

Any Hamiltonian cycle must use the edge (t, s)

Hamiltonian Cycle Problem

Claim: 3-SAT instance is satisfiable if and only if G has a Hamiltonian cycle.

Proof: \Rightarrow

Given a satisfying assignment for 3-SAT.

If $X_i = T$, traverse X_i gadget L to R, else R to L. Since each clause C_j is satisfied by the assignment, there has to be at least one path that moves in the right direction to be able to cover node c_j .

Hamiltonian Cycle Problem

Claim: 3-SAT instance is satisfiable if and only if G has a Hamiltonian cycle.

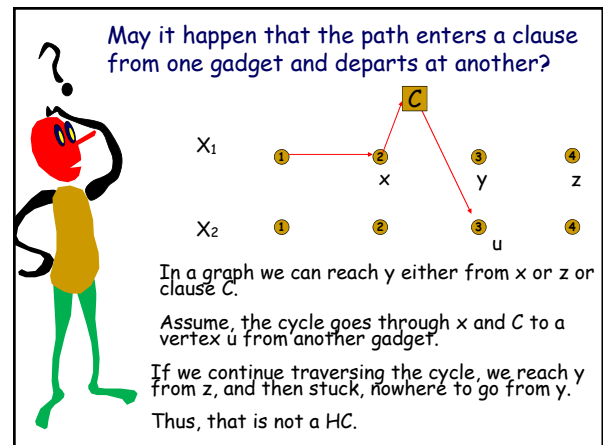
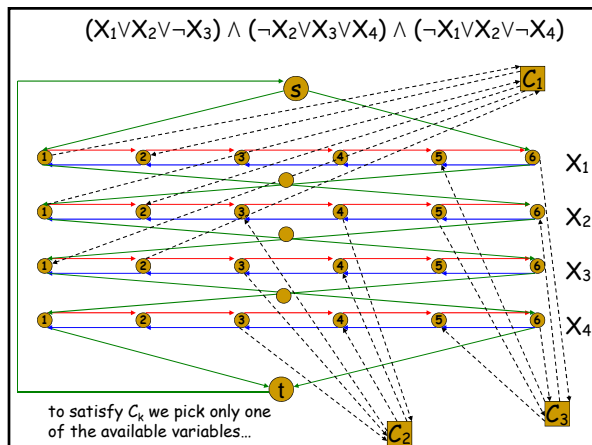
Proof: \Leftarrow

Given a Hamiltonian cycle.

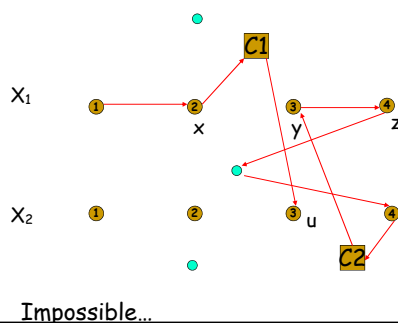
Set each X_i true if path goes L to R through X_i 's gadget, false if it goes R to L.

Do we satisfy all clauses?

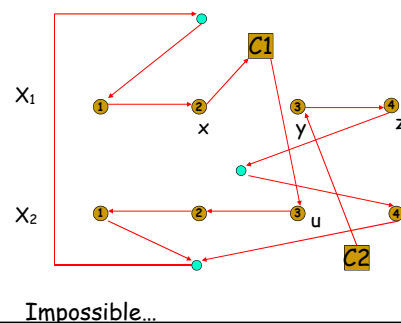
Consider any clause. We visit a clause in either LR or RL direction. If it's LR - X_i is true, so C_j is satisfied, since it contains X_i . If it's RL - X_i is false, so C_j is satisfied, since it contains $\neg X_i$.



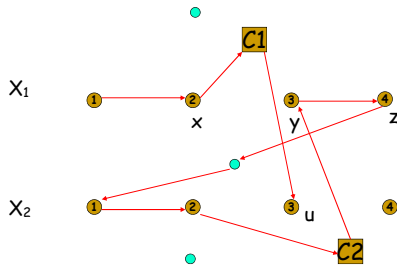
May it happen that the path enters a clause from one gadget and departs at another?



May it happen that the path enters a clause from one gadget and departs at another?

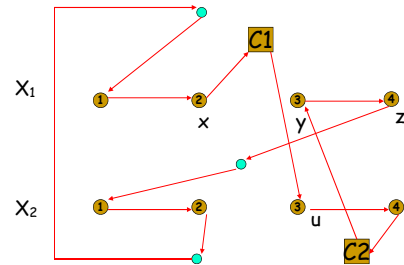


May it happen that the path enters a clause from one gadget and departs at another?



Impossible...

May it happen that the path enters a clause from one gadget and departs at another?



Impossible, since we enter C2 in a wrong direction...

Traveling Salesman Problem



Given a weighted complete graph $G=(V,E)$ with positive edge costs, find the shortest Hamiltonian cycle.

Is it in NP? No, it's not.

We cannot check that there is no shorter cycle in polynomial time.

It follows, TSP is not NP-complete.

TSP: decision version

Given a weighted complete graph $G=(V,E)$ with positive edge costs, is there a Hamiltonian cycle that has total cost $\leq k$?

Is it in NP?

Yes, we can verify the solution in polynomial time.

Decision TSP is NP-Complete

Proof by reduction from a HC.

Given the input $G=(V,E)$, we modify it to construct a complete graph $G'=(V',E')$ with cost on each edge as follows:

$c(u,v) = 1$, if edge $(u,v) \in E$
 $c(u,v) = 2$, otherwise.

Claim:

G has a HC of length k iff $|TSP(G')| = k = V$

Decision TSP is NP-Complete

G has a HC of length k iff $|TSP(G')| = k = V$

Proof.

\Rightarrow

By construction a cycle of length k has the total cost of k . Here k is the number of vertices.

Decision TSP is NP-Complete

G has a HC of length k iff $|TSP(G')| = k = V$

Proof.

\Leftarrow)

There is a tour that visits every vertex once with weight at most k .

Since $k=V$, every edge traversed must have weight 1.

Thus, this will create a HC of length k .

Don't be afraid of NP-hard problems.



Many reasonable instances (of practical interest) of problems in class NP can be solved!

The largest solved TSP an **85,900-vertex** route calculated in 2006. The graph corresponds to the design of a customized computer chip created at Bell Laboratories, and the solution exhibits the shortest path for a laser to follow as it sculpts the chip.