We will discuss

- Solving NP complete problems

Detecting NP completeness

→ Take a known np-complete problem and reduce it to said problem.

What do you do when you can't find polynomial-time algo for a problem?

→ Do some research
→ Try to show NP-completeness
→ Discuss with peers
→ Settle for a sub-optimal algorithm if the stakes aren't high

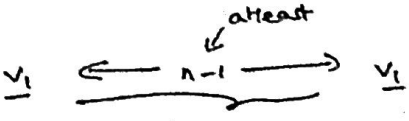
We will focus on 2nd technique.

1. Show problem X is in NP.
2. Finding a reduction from Y to X (Y is known to be np complete)
   ↳ polynomial time

→ Step 1 is relatively easier.

Consider a problem: the shortest tour problem. Given a weighted undirected graph $G(V, E)$. Start from a vertex V_1 and find shortest path such that you visit all the vertices in V and ends in V_1. (Traveling Salesman Problem)

brute force: try all possible paths



possible paths $\geq (n-1)!$

$O(n!)$ ⇒ worse than exponential

Step 1: Show TSP is in NP

Form TSP as a decision problem

- Given a graph and a start vertex, is there a tour of length at most d?

Idea: non determinism can guess which vertex to be next at each step

(also)

once shortest tour has been found, it can be verified in polynomial time on deterministic RAM

✓ step 1

Step 2: Show any input for a NP complete problem can be transformed in poly. time to an ip. for TSP.

Let the np-complete problem be SAT

Reducing SAT to TSP (idea: compare & contrast)

SAT:

IP: A Boolean Formula

OP: Does it have a satisfying assignment

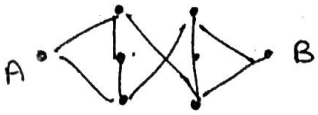
- Each variable is either True or False
- Deciding value for each variable is hard
- Each clause must be satisfied
- One variable is enough to satisfy a clause

TSP

IP: Graph with distances and d.

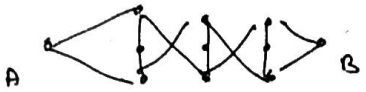
OP: Is there a tour of length $\leq d$

- Every city visited at least once
- The main decision is the order of vertices
- Can visit a city more than once, not necessary
- Don't visit a city more than once on purpose

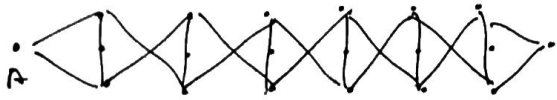


shortest path length = 7
from (A->B), covering all vertices

different shortest paths = 2

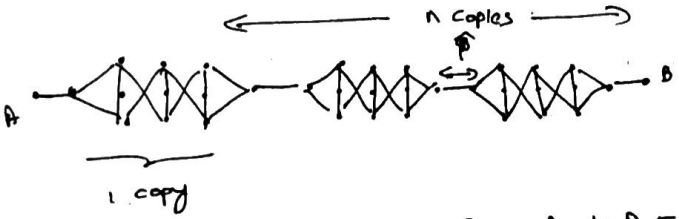


#shortest paths = 2
length = 10

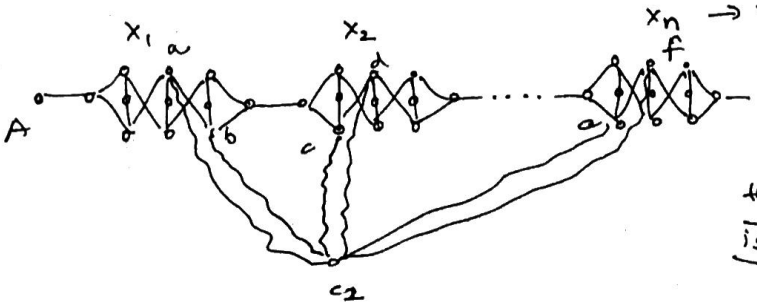


#shortest paths = 2
length = 19

idea: each possible path can be "true" or "false"
if we go up (lets call it true)
false, otherwise



shortest paths from A to B = 2^n



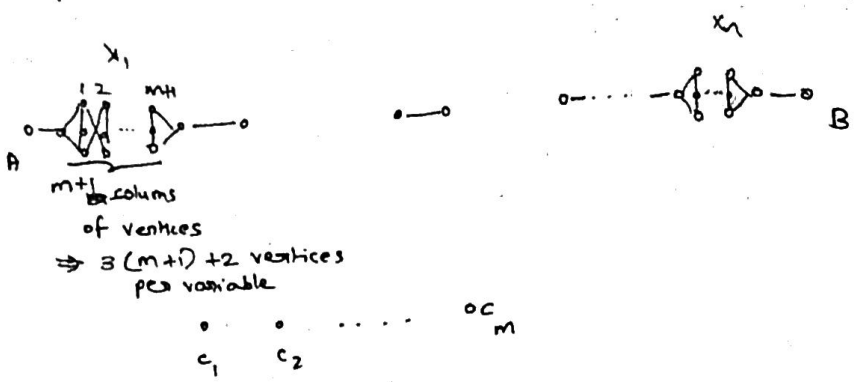
→ Variables in boolean formula

~~let's say~~ only x_2 is present in clause c_1

let $c_1 = (\bar{x}_1 \vee x_2 \vee x_n)$

c_1 has to be visited from a to b in x_1 ⇔ x_1 is false
or from c to d in x_2 ⇔ x_2 is true
or from e to f in x_n ⇔ x_n is true

SAT ip: n boolean variables, m clauses



vertices for variables $n(3(m+1)+2)$

$3mn+3n+2n$

$5n+3mn$

vertices for clauses = m

others = 2

total = $5n+3mn+m+2$

If boolean formula has satisfying assignment what is the shortest path's length?

foreach clause vertex, we pay 1 (shortest path case)

from A to B without clauses $\Rightarrow$ $5n+3mn+1$

total = $5n+3mn+m+1$

↓

each vertex is visited exactly once

* add one edge b/w B → A (complete the tour)

shortest length = $5n+3mn+m+2$

$(3(m+1)+1)n+n+1$ $(3(m+1)+1)n+(n+1)+1$

$(3m+3+1)n+n+2$

$3mn+4n+n+2 = 5n+3mn+2$

$3mn+4n+n+1 = 3mn+5n+1 \Rightarrow$ length w/o clause vertices

Library of NP Completeness → huge list of problems shown to be np-complete.

Richard Karp in 1972 showed ~20 problems to be NP complete

exercise: convert the boolean formula

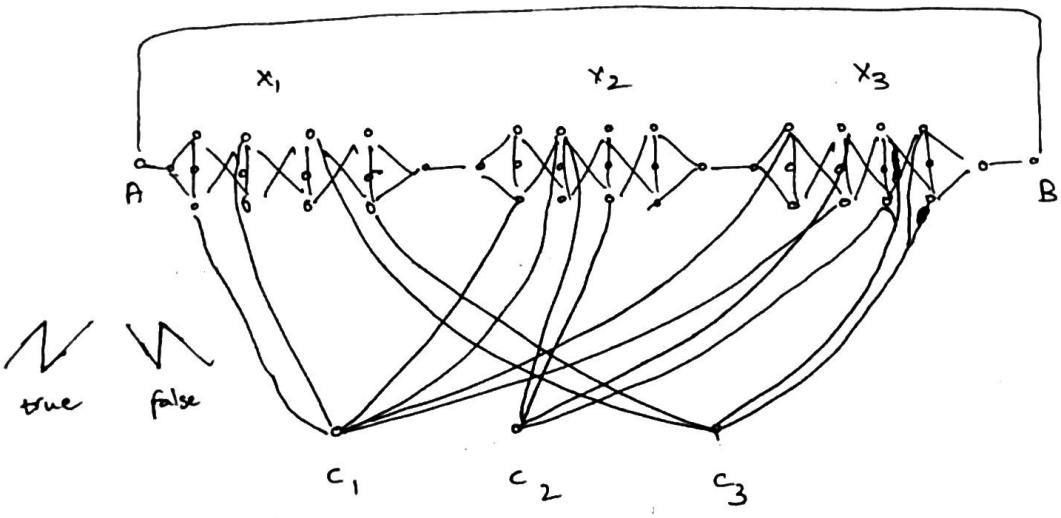
$$(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_1)$$

into a shortest path problem
and therefore find its satisfiability

---

i/p → $n = 3$

$m = 3$



to satisfy the formula we need the shortest ~~tour~~ tour to be of length $5n + 3mn + m + 2$ $(A \to A)$

only c_1 or c_3 can be visited in x_1

only c_1 or c_2 can be visited in x_2

only $(c_1 \& c_2)$ or c_3 can be visited in x_3

c_1 in x_1, c_2 in x_2 and c_3 in x_3

⇒ every vertex visited only once

∴ shortest length $= 5n + 3mn + m + 2$ *

each Variable its $3(m+1)+1 = 13$

and for ~~the~~ each clause, ~~it~~ its 1 ~~$\Rightarrow 3(m+3)$~~

~~$(3(m+1)+1+1)n$~~

$= (3m+3+2)n = 5n+3mn$

$(13+3)\cdot 3$

↓ one variable ↓ # variables

$= 16 \cdot 3 = 48$

So per Variable $\Rightarrow (13+1)\cdot n$

$= 14.n$

$= 14.3$

$= 42$

Each connector $\Rightarrow 1$

connectors $= n+1 = 4$

from $B \to A \Rightarrow 1$

5

total $= 42+5 = 47$

$5n + 3mn + m + 2$

$15 + 27 + 3 + 2$

$20 + 27$

47

∴ The boolean formula is satisfiable

Border blw P and NP-complete

eg: 3SAT is NP-complete, while 2SAT is in P ($n=3$)

the border is so fragile, that we need to be rigorous while showing np completeness.

ggwp.