Walchand College of Engineering

(Government Aided Autonomous Institute)

Vishrambag, Sangli. 416415

Computer Algorithms

Complexity classes, NP and Approximate Algorithms

24-25

D B Kulkarni

Information Technology Department

Algorithm Classification based

SN	Criteria	Type	Y/N	Algorithm Title	Remark
1	Processing	State-transition	Yes	Deterministic	Goes through same state every time it is run?
			No	Nondeterministic	
		Concurrency	Yes	Parallel	Execution order
			No	Sequential	
		Randomizer	Yes	Randomised	Uses random number generator?
2	Output	Binary	Yes	Decision	Output 0/1?
			No	Optimization	
		Estimate	Yes	Approximate	Output is near optimal
3	Run time	Polynomial	Yes	Polynomial	
			No	Non Polynomial?	

Deterministic Vs Non- Deterministic

- Deterministic: If machine (H/W) goes through same state every time the program is run?
- Non-Deterministic- Everytime program is run the state through which m/c goes will be different

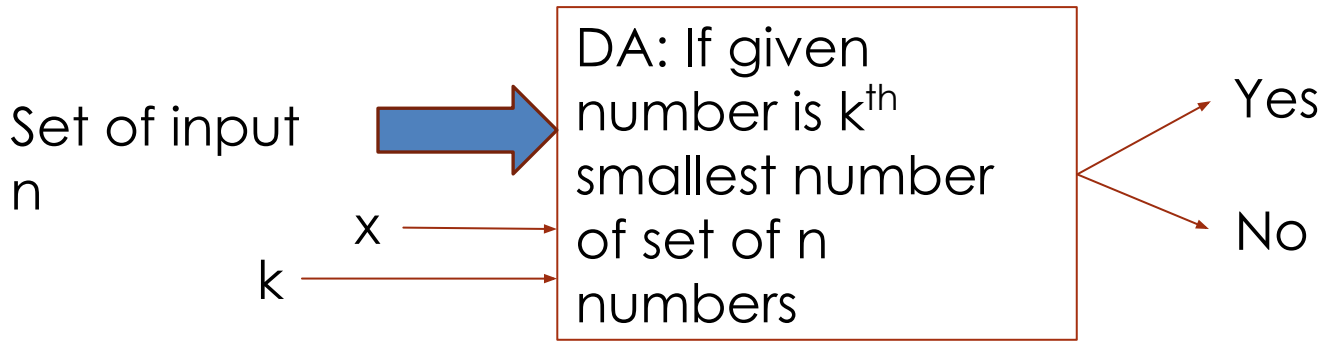
Decision Vs Optimization

- Decision – Output – Yes/No, True/False, 0/1
- Optimization- Output based on some optimization criterion
  - Example: Knapsack, sorting, Matrix multiplication

- Complexity: Decision problems are simpler than optimization problems
- Can we express optimization problems in terms of decision problems?

Optimization Algorithms (OA) in terms of Decision Algorithms (DA)

- Sorting



Optimization Algorithms (OA) in terms of Decision Algorithms (DA)

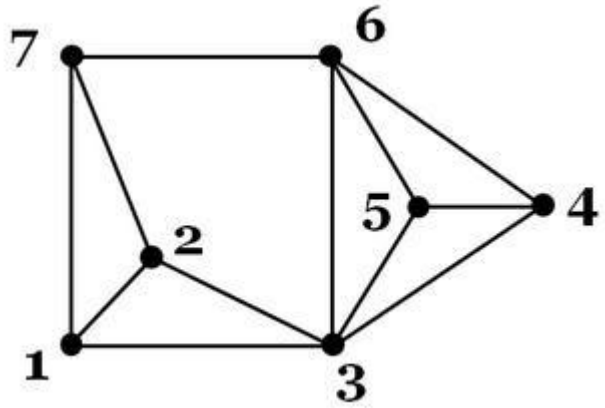
- Knapsack? Fractional, 0/1 ??

Some Problems

- Clique: A complete subgraph of a given graph
- Circuit satisfiability (CKT-SAT): Given combinational ckt (composed of AND, OR, Not gates), is it satisfiable? (produces output as 1)
  - Satisfiability (SAT): A Boolean formula with satisfying assignment (produces o/p 1 for some Boolean input assignment) is satisfiable
  - 3 Conjunctive Normal Form of Satisfiability (3 CNF SAT): Boolean expression in which every clause has exactly 3 literals
- Vertex cover (VC): Subset of vertices which covers (all edges of graph are coincident on at least one of the vertex in the VC) all the edges of graph.
- Hamiltonian cycle (HAM-CYCLE): Simple cycle (every vertex is visited exactly once)that contains each vertex in V.
  - Travelling salesman Problem (TSP): TSP builds on the HCP and is concerned with computing the lowest cost Hamiltonian cycle on a weighted digraph.

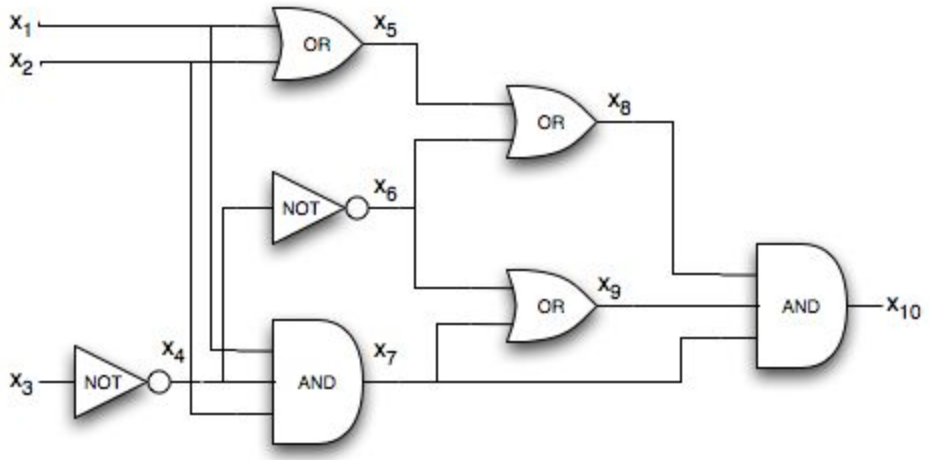
Clique

- A complete subgraph of a given graph
  - Cliques {1,2,7},{1,2,3}…. {3,4,5,6}
  - Max clique is no of elements in biggest cliques i.e. in {3,4,5,6} which is 4



Circuit satisfiability (CKT-SAT)

- Given combinational ckt (composed of AND, OR, Not gates), is it satisfiable? (produces output as 1)
  - For 110 input it produces output as 1



Satisfiability

- Boolean function:
  - A Boolean formula with satisfying assignment (produces o/p 1 for some Boolean input assignment) is satisfiable
  - F=ABC'+AB'C'D+ACD+AC
    - Variable: e.g. A, B, C
    - Literal: e.g.A, C'
    - Clause: Term e.g. ABC'
    - **True for A=B Or C Or D=1**

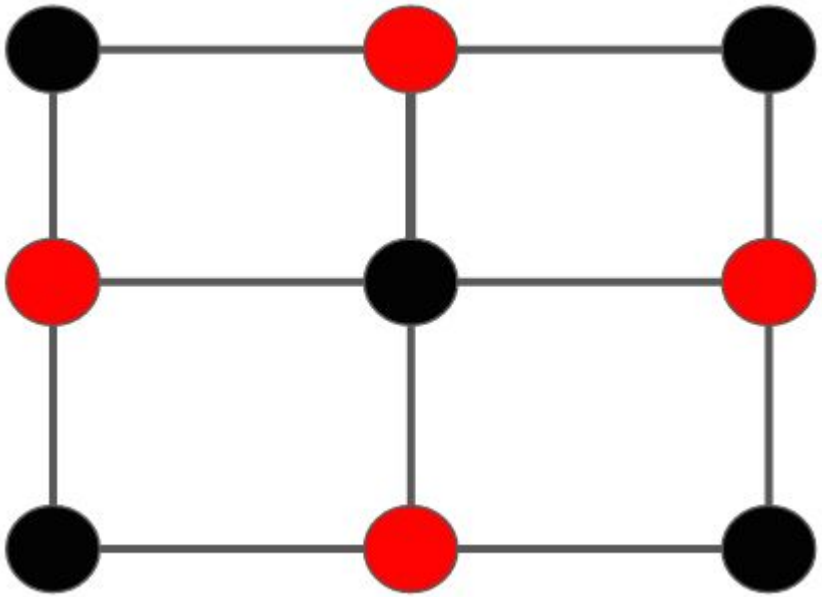
3 Conjuctive Normal Form Satisfiability (3CNF-SAT)

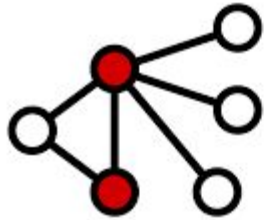
- 2 CNF: A Boolean expression in CNF (product of Sum POS) in which every clause has exactly 2 literals

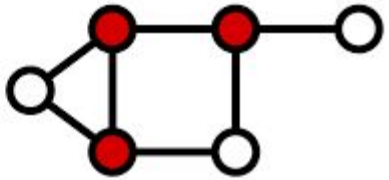
- A Boolean expression in CNF (product of Sum POS) in which every clause has exactly 3 literals
  - E.g. X= (A+B+C). (A+B+D). (A+B+C')
  - clause e.g. (A+B+D) and literal e.g. A, C'

Vertex cover (VC)

- Subset of vertices which covers (all edges of graph are coincident on at least one of the vertex in the VC) all the edges of graph.

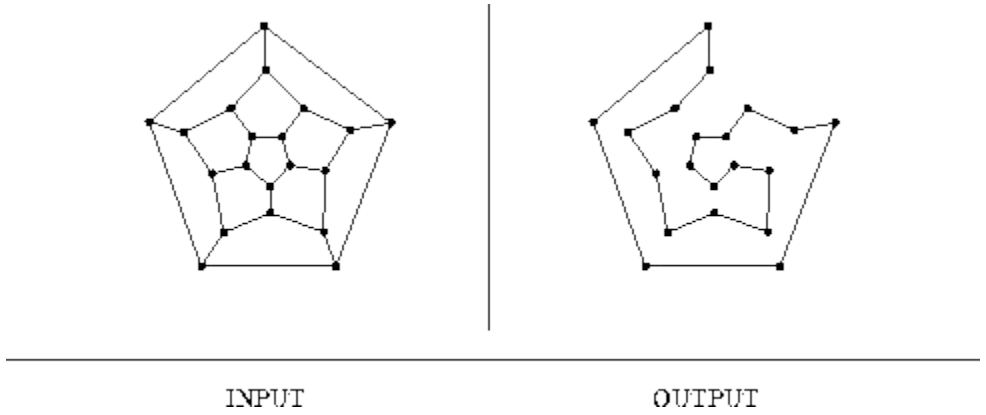




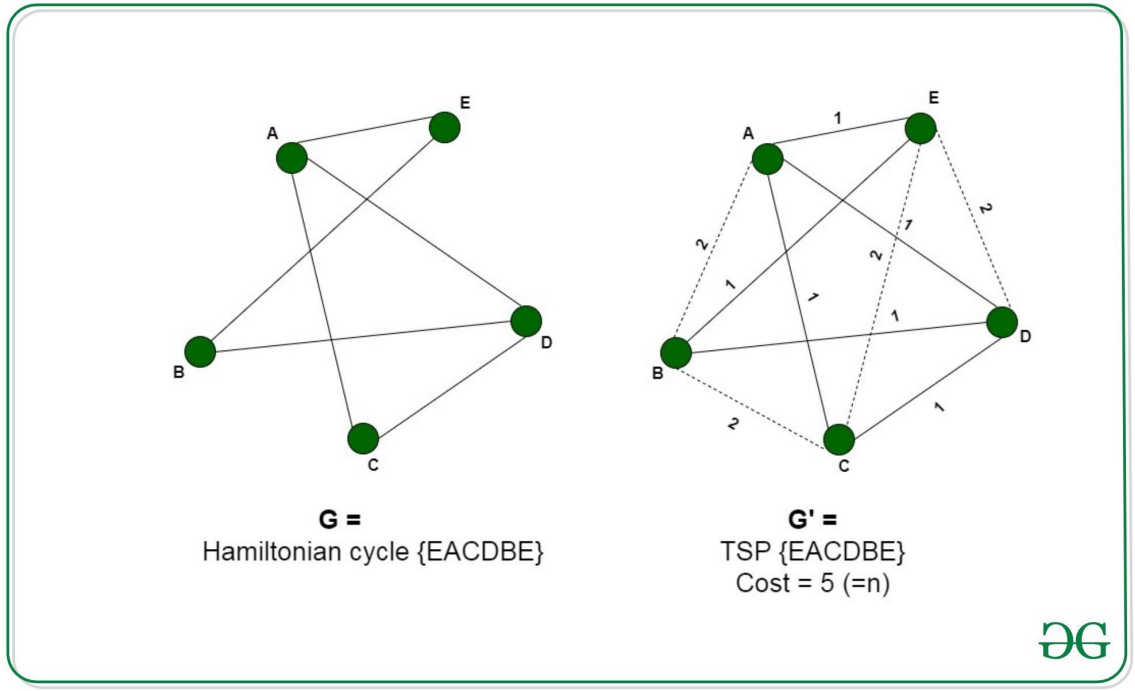


Hamiltonian cycle Problem(HCP/ HAM Cycle)

- Simple cycle (every vertex is visited exactly once)that contains each vertex in V (all vertices in given graph).



- Built on HCP where we found shortest HCP
- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?



Decision problems

- Decision problems are much simpler than optimization problem
- If decision problem can't be solved in P time, then its optimization counterpart can also not solved in P time
  - If decision problem can be solved in P time, then its optimization counterpart may be solved in P time

Let's consider *decision problems* hereafter

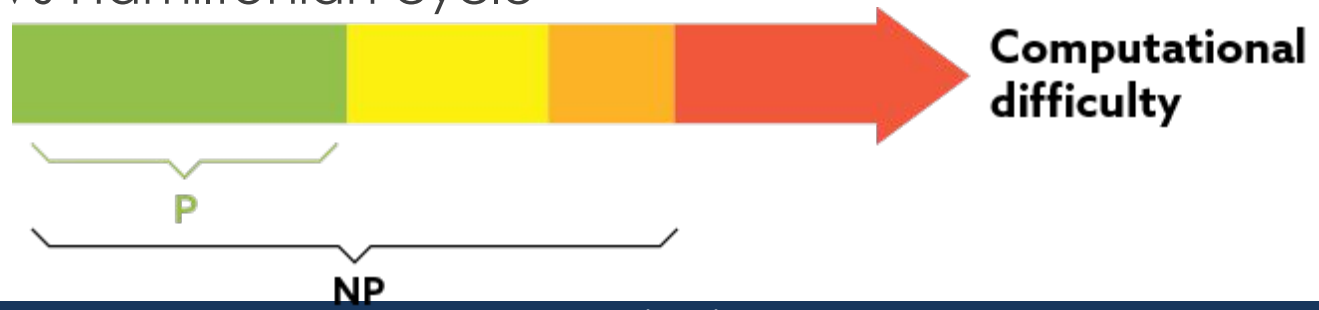
Polynomial (P)

Problem can be solved in polynomial time by deterministic time machine (instance can be verified in polynomial time $O(n^q)$)

- Solution or certificate is verifiable in polynomial time

Non-Deterministic Polynomial (NP)

- Instance of the Problem can be verified (for yes) in polynomial time by deterministic time machine (DTM)
- Problem can not be solved in polynomial time by deterministic time machine
  - Problem can be solved in polynomial time by non-deterministic time machine (NTM)
- Complexity $O(a^n))$
- Similar problem: one in P and other in NP
  - Shortest {O(V.E)} Vs Longest path
    - Finding longest path is difficult
  - Finding whether graph contains simple path with at least given no. of vertices is in NP
  - Euler Vs Hamiltonian cycle

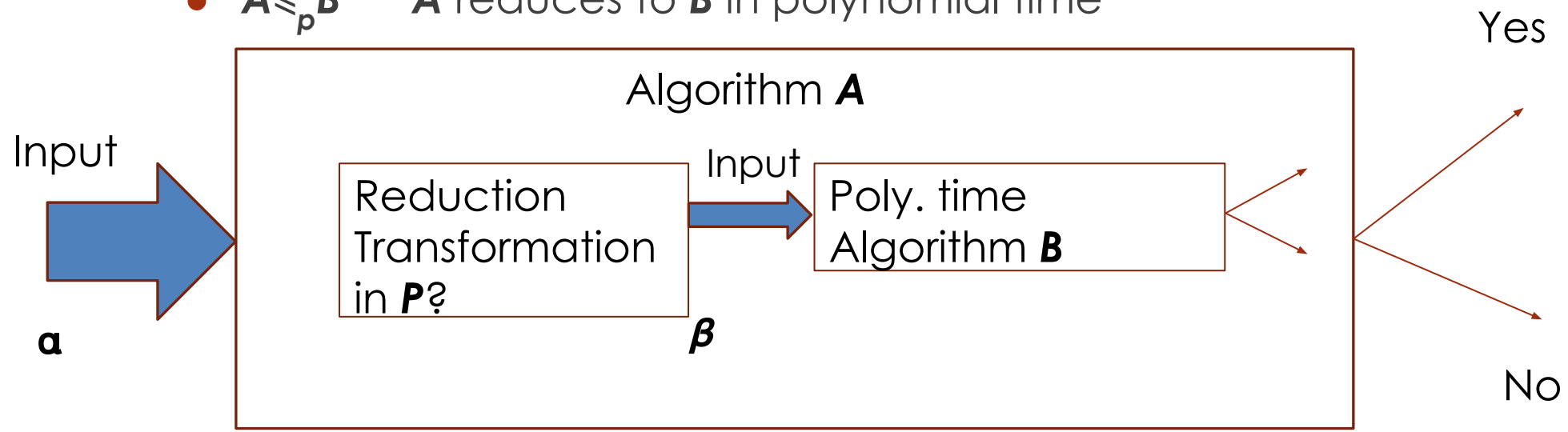


P=NP?

- Is P proper subset of NP?
  - Proper subset- At least one element in NP which is not in P
- Clay Mathematics Institute is offering **a US$1 million prize to anyone who can prove that P = NP?**

Reductions

- Its mapping between the problems.
- Can we solve problem ***A*** using solution of problem ***B***?
- Used to show that the unknown problem can/not be solved in P time?
  - Let ***A*** be the problem to be solved (Algorithm ***A***)
  - Let ***B*** be problem with known solution which is in ***P***
  - If we can express (solve) ***A*** in terms of ***B***
    - $A \leqslant_p B$ ***A*** reduces to ***B*** in polynomial time

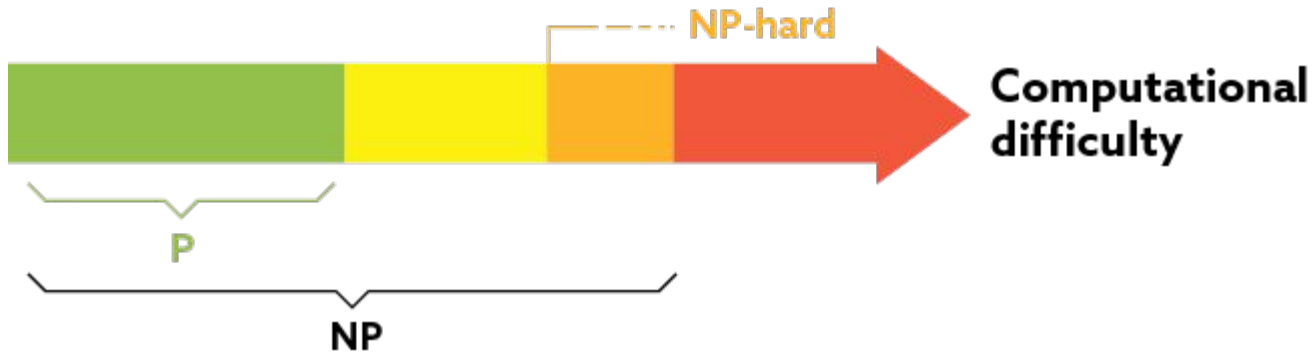


- If Reduction can be carried out in polynomial time
  - If B is polynomial then A is also polynomial

NP Hard (NPH)

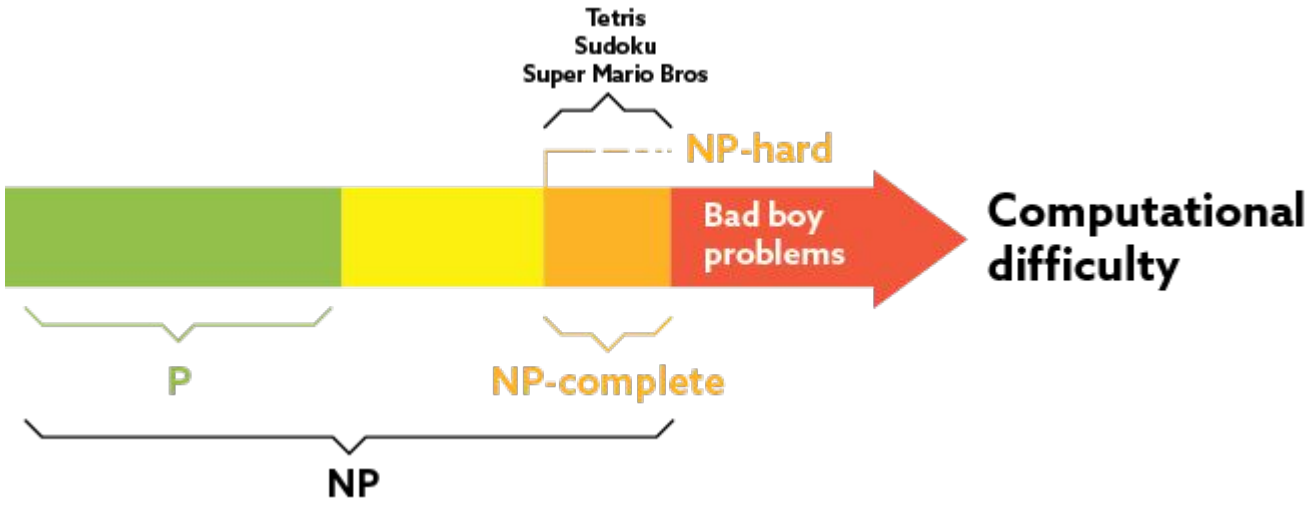
- Non deterministic polynomial time hard (**NPH**): at least as hard as the hardest problems in NP. Examples: Halting problem, subset sum

  - If a problem **X** is NPH, then every problem A,B,C… in NP can be reduced to X in polynomial time.
  - can be solved in polynomial time then all NPC problem can be solved in polynomial time.



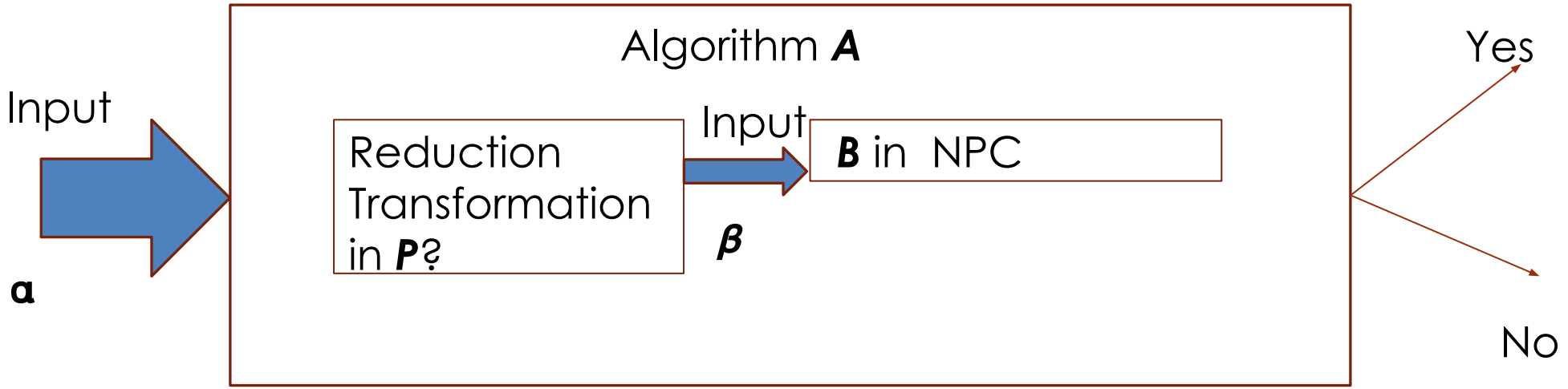
NP Complete (NPC) Problem

- A problem in NPC has a property that if it can be solved in polynomial time then all other problems in NP can also be solved in polynomial time.
- It is easy to show that a problems in NP can be reduced to NPC.
- All NP Complete problems are NP Hard but some NP Hard problems are **not** known to be NP Complete



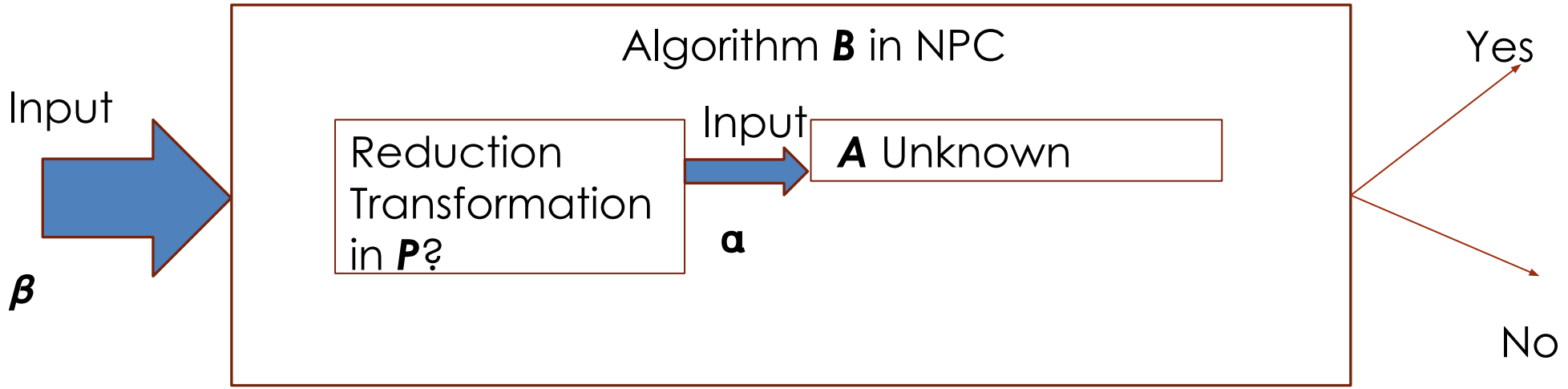
Reductions for NPC

- Compare problem with known NPC
- Let A be the problem to be solved (Unknown)
- Let B be problem which is known tobe in NPC
    - A belongs to NPC if
        - we can express (solve) A in terms of B (A can be solved using B's solution)
        - Inputs of A can be transformed into inputs of B in polynomial time

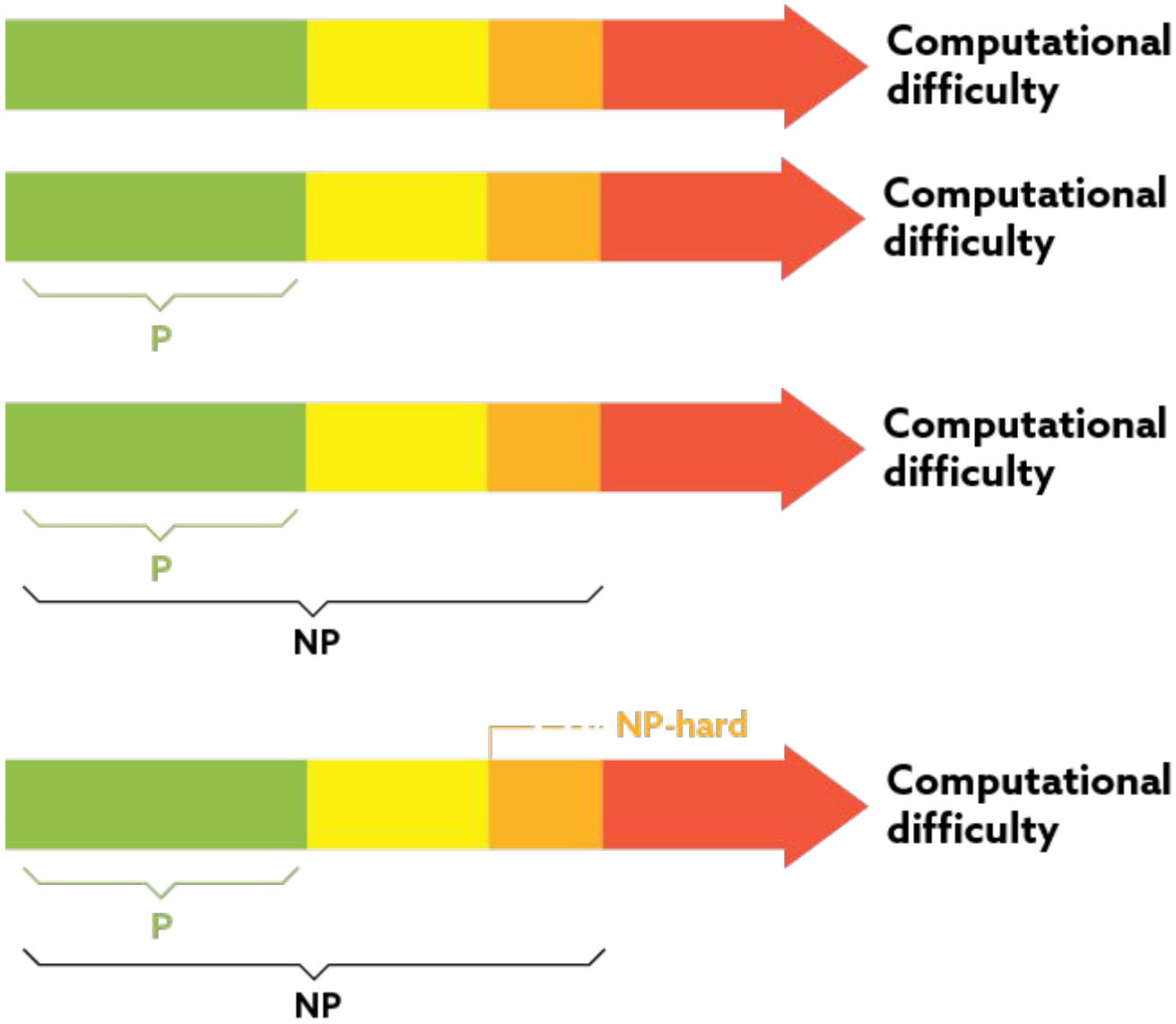


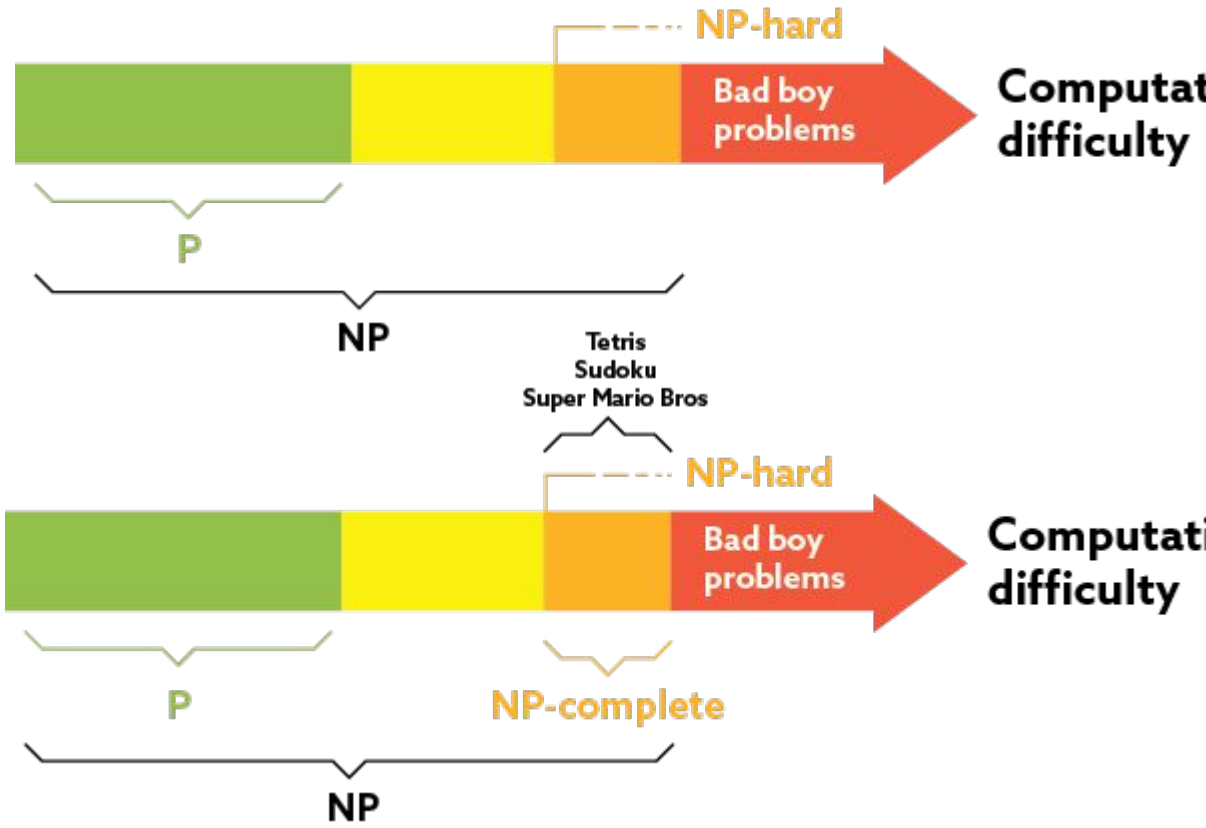
Reductions for NPC?

- Compare problem with known NPC
- Let A be the problem to be solved (Unknown)
- Let B be problem which is known tobe in NPC
    - A belongs to NPC if
        - we can express (solve) A in terms of B (A can be solved using B's solution)
        - Inputs of A can be transformed into inputs of B in polynomial time

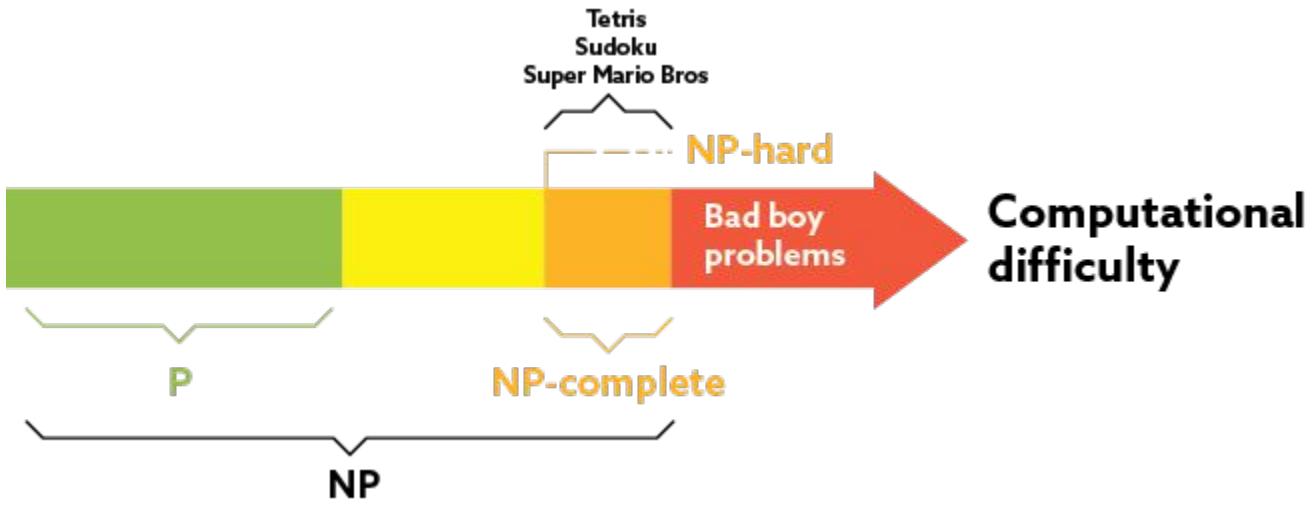


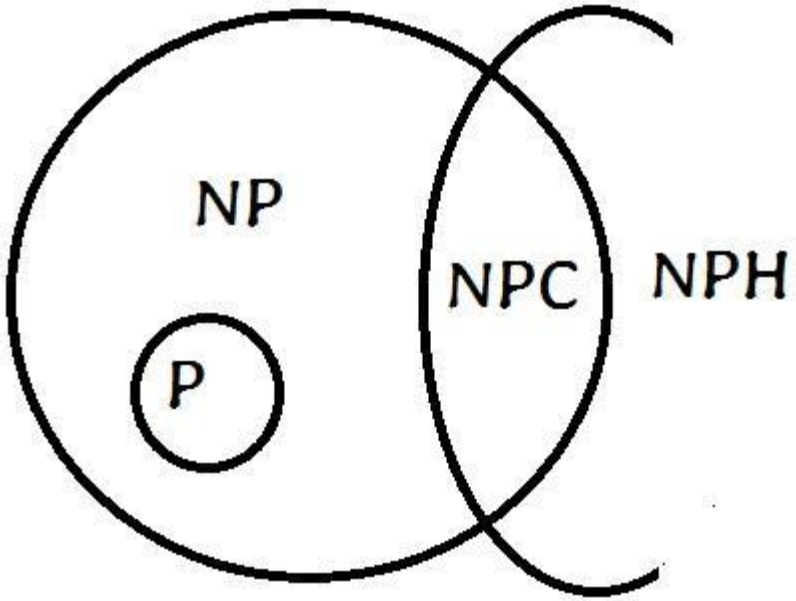
P Vs NP Vs NPC Vs NPH





Classification





- NP
- P
- NPC
- NPH

Proof of : ***X*** belonging to NPC

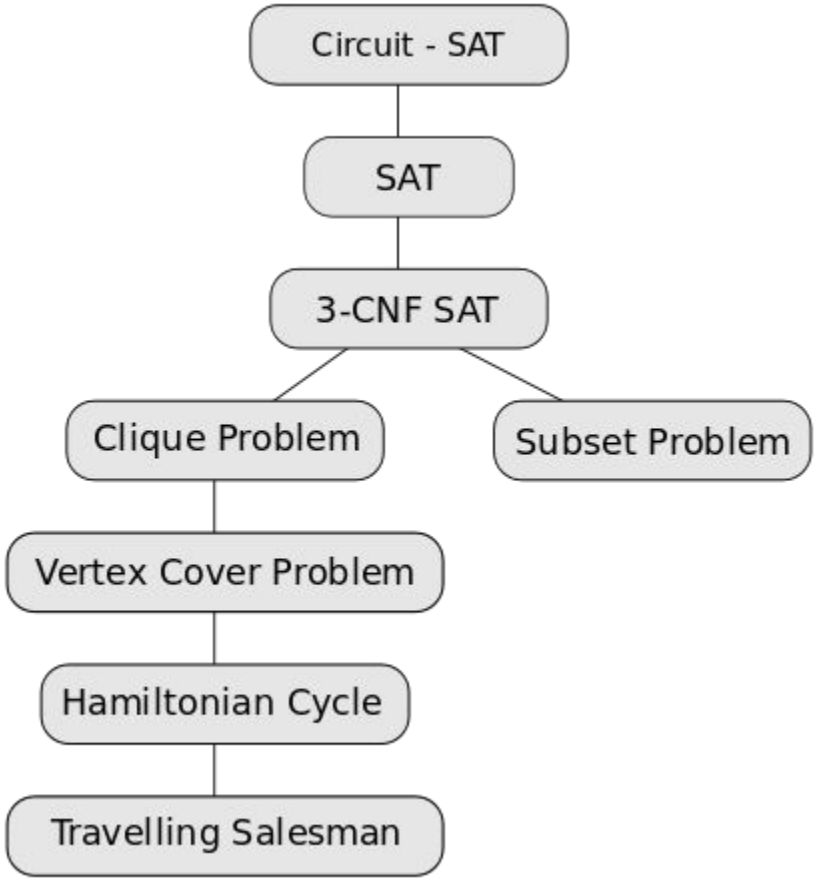
X is unknown problem, Y belongs to NPC

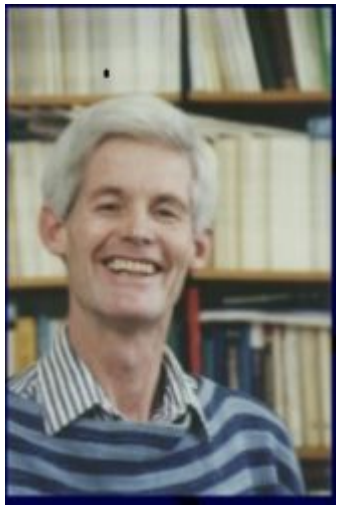
1. Show that problem **X** belongs to **NPC**
2. Show that problem ***X*** can be expressed in terms of ***Y***
3. Show that instance of problem ***X*** can be converted into instance of ***Y*** in **polynomial time**

Assume there is a problem (***Y***) which is already proven to belong to **NPC**

- We take unknown problem (***X***) and show that it is as hard as ***Y***. ***X*** can be mapped on ***Y*** (***X*** can be reduced to ***Y***) in polynomial time
    - Other problem (***Z***) can be mapped on ***X*** and so on……
    - If ***X*** belong to **NPC**, ***Z*** problems also belong to **NPC**
    - Problems ***X,Z*** ….belongs to **NPC**

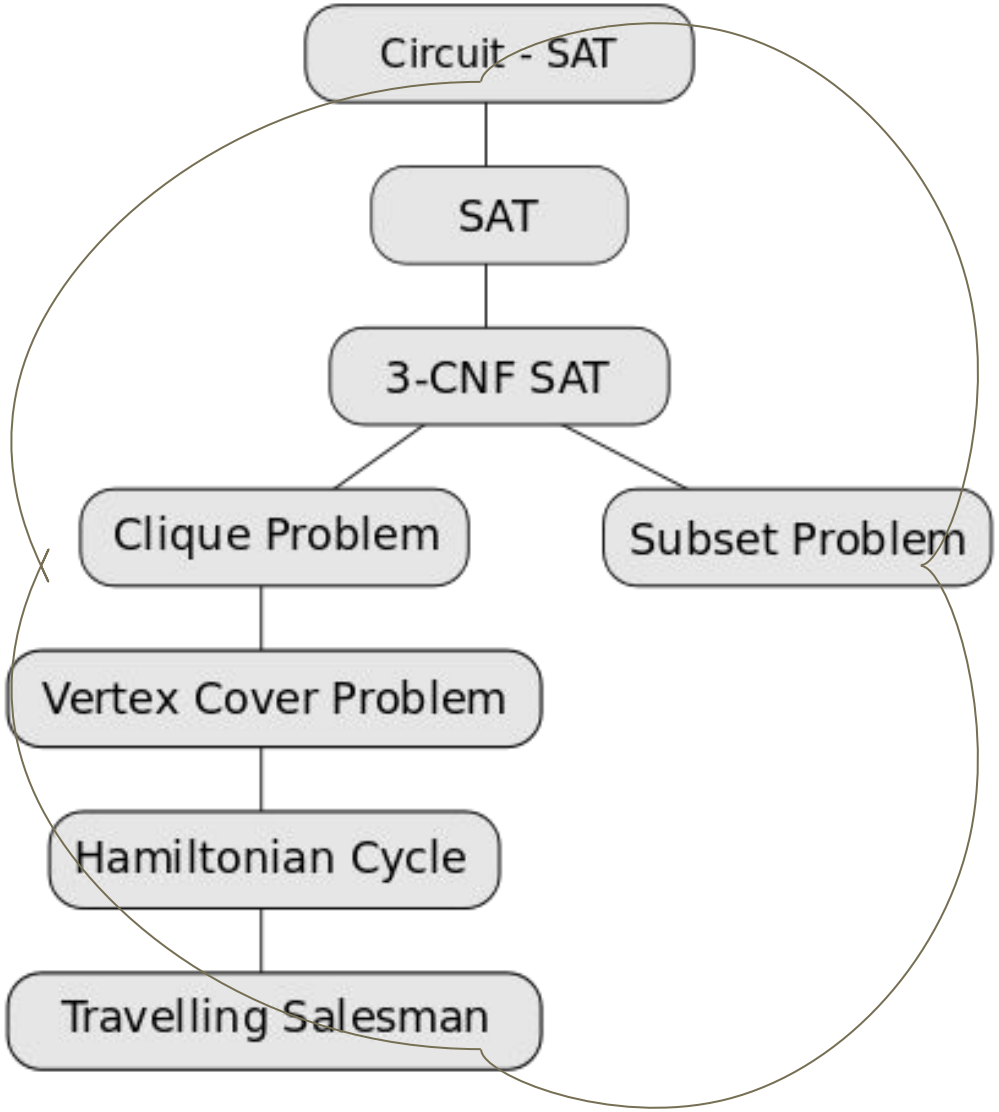
NP complete problems



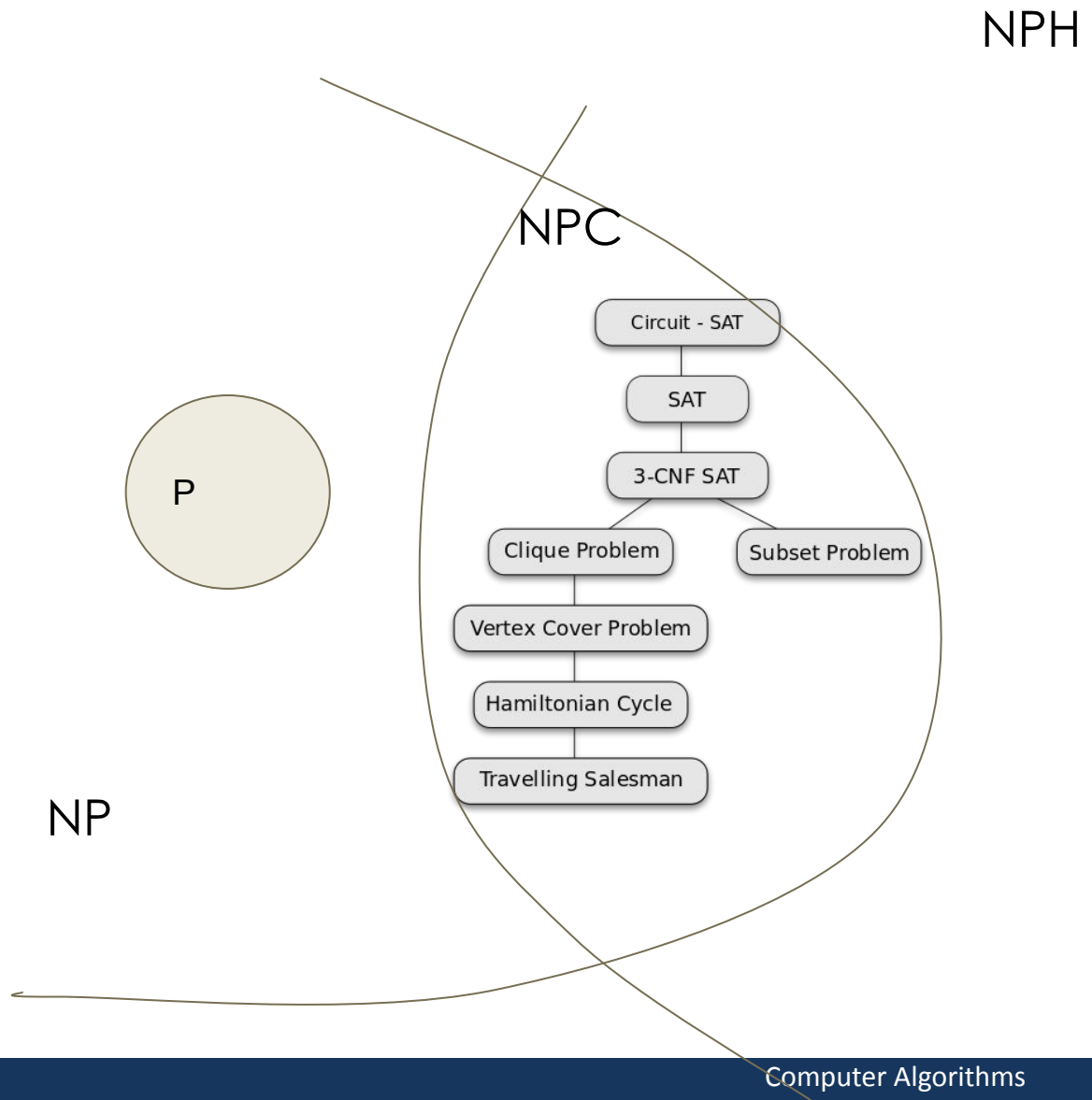


Shepen Cook (www.cs.toronto.edu)
SAT Problem (first) is NP Complete

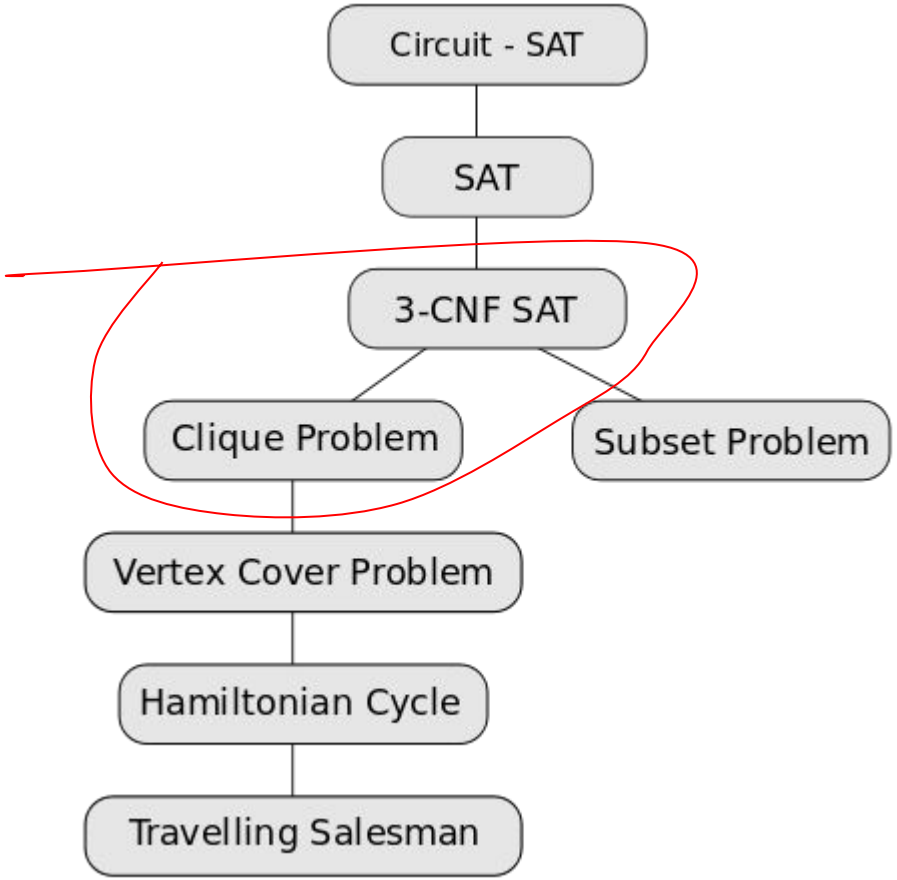
NP complete problems



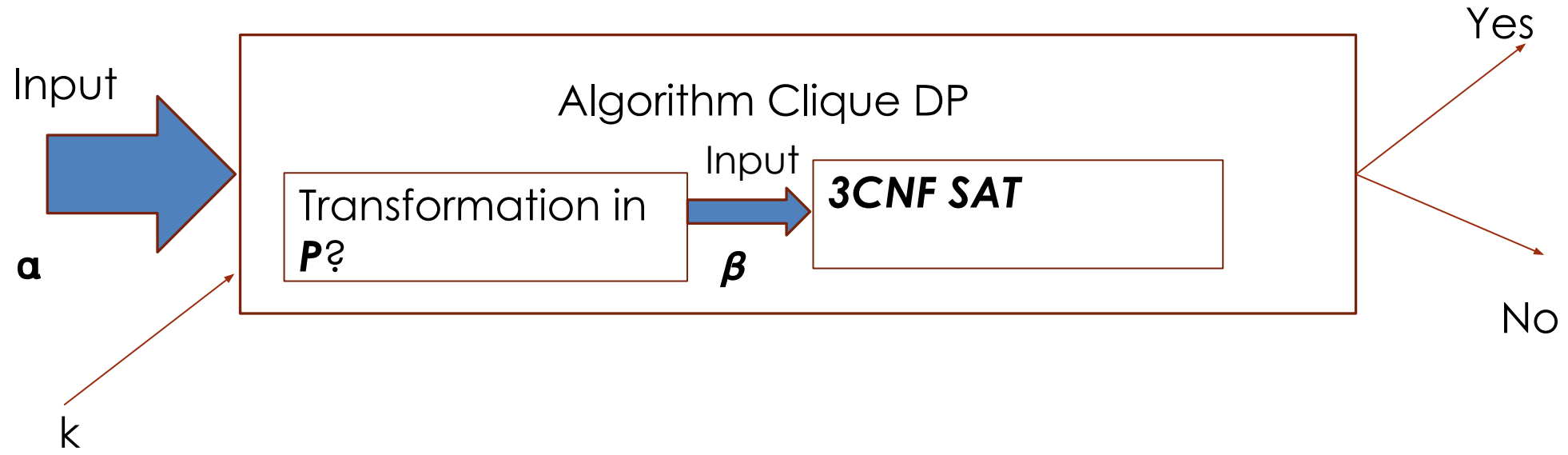
NP complete problems



NP complete problems



Clique decision Problem(CDP) is in NPC



Clique decision Problem(CDP) is in NPC

CDP: Does graph has **clique** of size **k**?
Given : 3CNF SAT belongs to NPC

Steps:

1. Show that CDP belongs to NP: Given the certificate, it can be verified in polynomial time.
    a. Certificate: Set S consisting of k nodes in the clique and S is a complete subgraph of G
    b. Each vertex in S can be checked if it is connected to other k-1 vertices (requires O(k^2)time)
2. Show that CDP can be mapped on 3CNF SAT in polynomial time

Clique decision Problem(CDP) is in NPC

1. Show that CDP can be mapped on 3CNF SAT in polynomial time

- Take boolean equation, in 3 CNF form

$$F=(X_1+X_2+X_3')(X_1'+X_2'+X_4)(X_2+X_3+X_4')\ldots\ldots \text{ (in 3 CNF form)}$$

with 3 variables, 3 clauses

Construct a graph

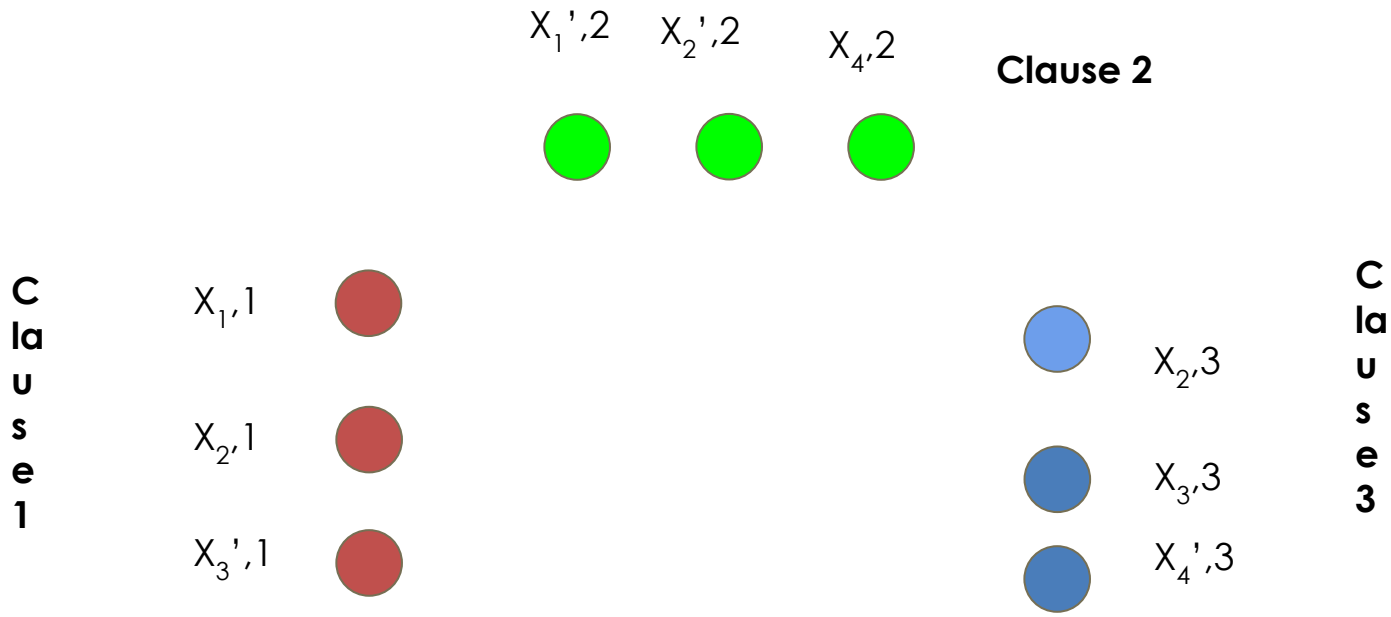
1. Every clause represents a group of vertex
2. Every literal represents a vertex
3. Add edges in the graph between vertices of different group provided
   a. The two vertices belong to different group
   b. Vertices are not complement of each other

Clique decision Problem(CDP) is in NPC

$F=(X_1+X_2+X_3')(X_1'+X_2'+X_4)(X_2+X_3+X_4')$..... (in 3 CNF form)

Construct a graph

1. Every literal represents a vertex
2. Every clause represents a group of vertex



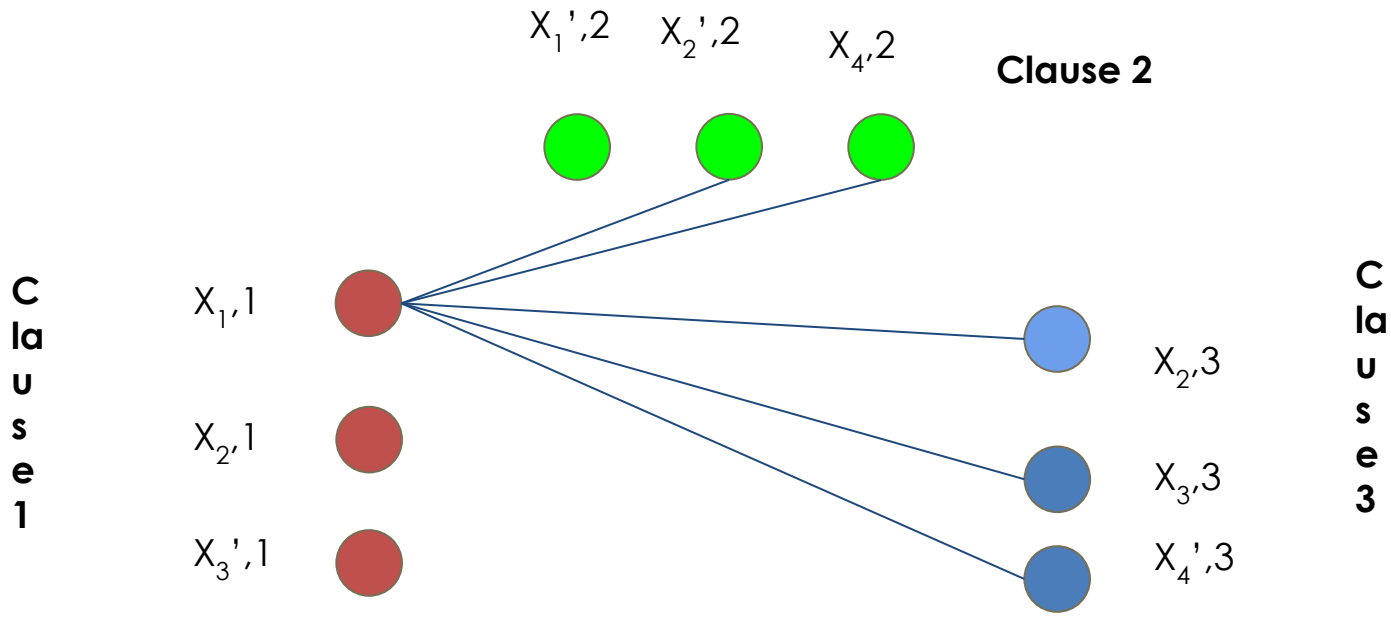
Clique decision Problem(CDP) is in NPC

$F=(X_1+X_2+X_3')(X_1'+X_2'+X_4)(X_2+X_3+X_4')$..... (in 3 CNF form)

Construct a graph

Add edges in the graph between vertices of different group provided

a. The two vertices belong to different group
b. Vertices are not complement of each other



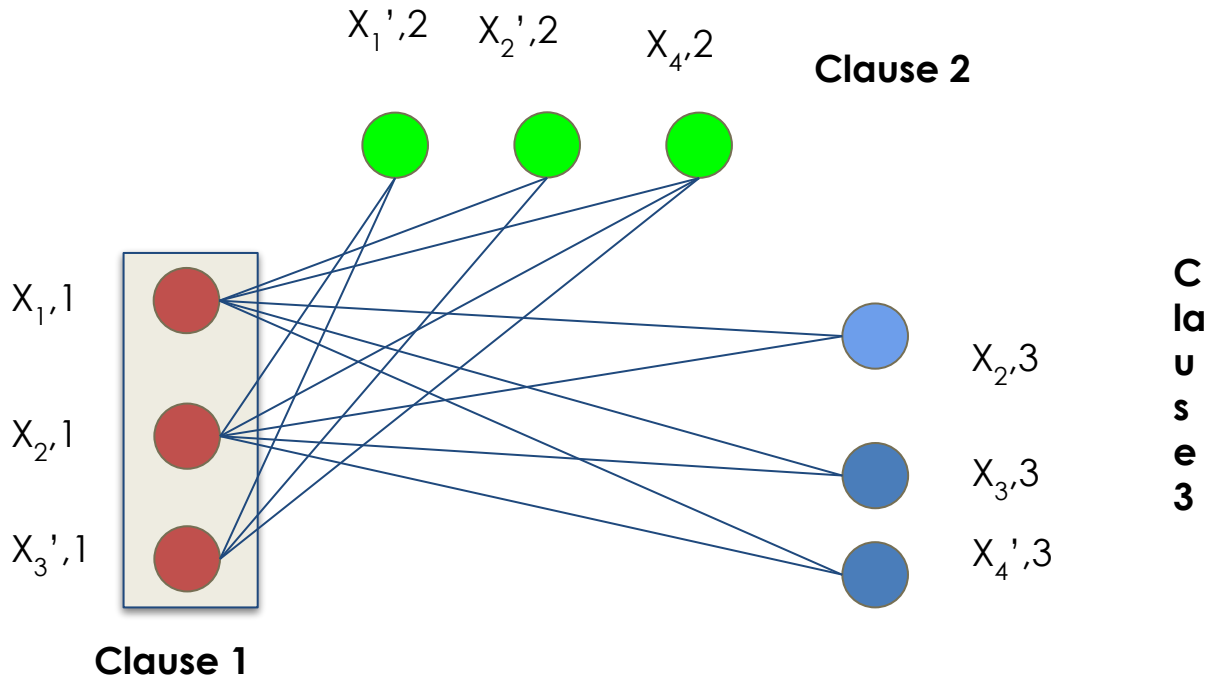
Clique decision Problem(CDP) is in NPC

$F=(X_1+X_2+X_3')(X_1'+X_2'+X_4)(X_2+X_3+X_4')$..... (in 3 CNF form)

Construct a graph

Add edges in the graph between vertices of different group provided

a. The two vertices belong to different group
b. Vertices are not complement of each other

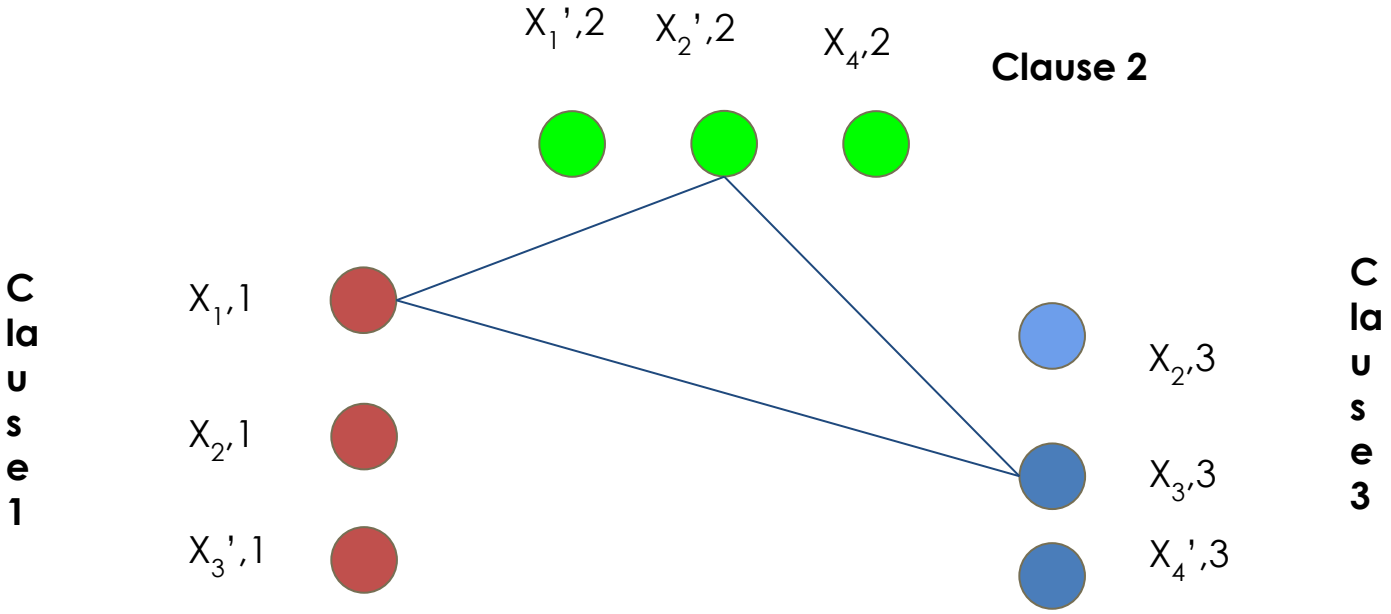


Clique decision Problem(CDP) is in NPC

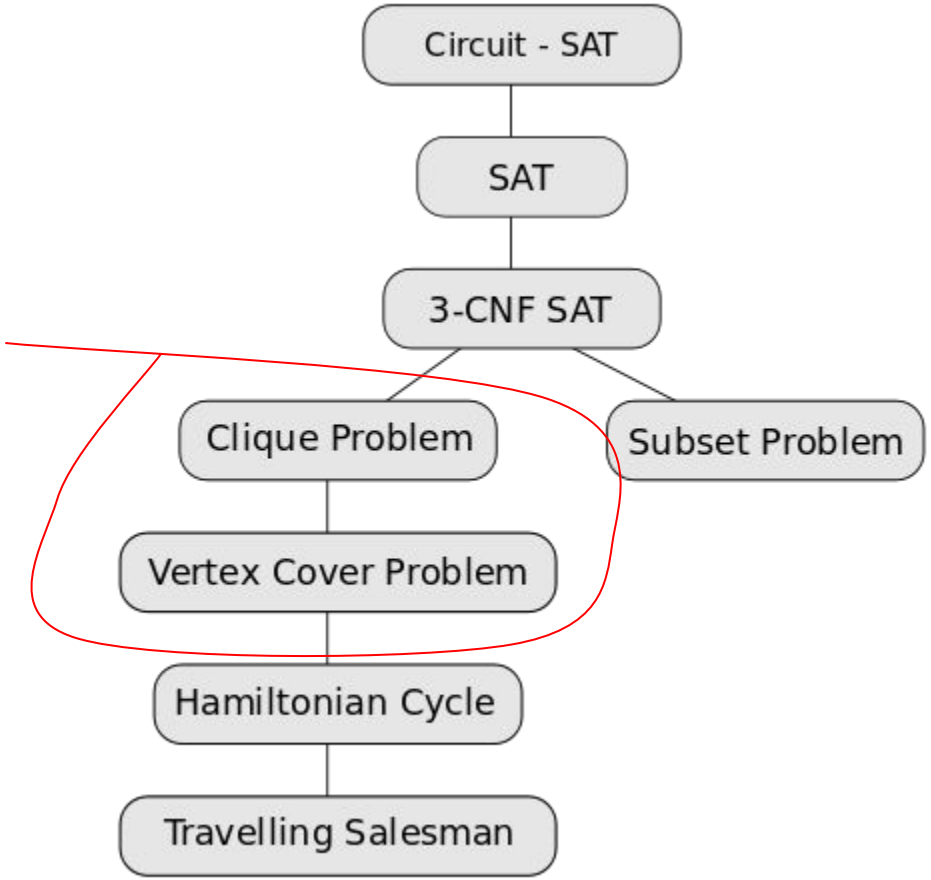
$F=(X_1+X_2+X_3')(X_1'+X_2'+X_4)(X_2+X_3+X_4')$..... (in 3 CNF form)

The 3-CNF Satisfiability instance reduced to Clique. The Red (X_1,1), Green (X_2') and Blue (X_2,3) vertices forms a 3-clique and correspond to one of the satisfying assignment.

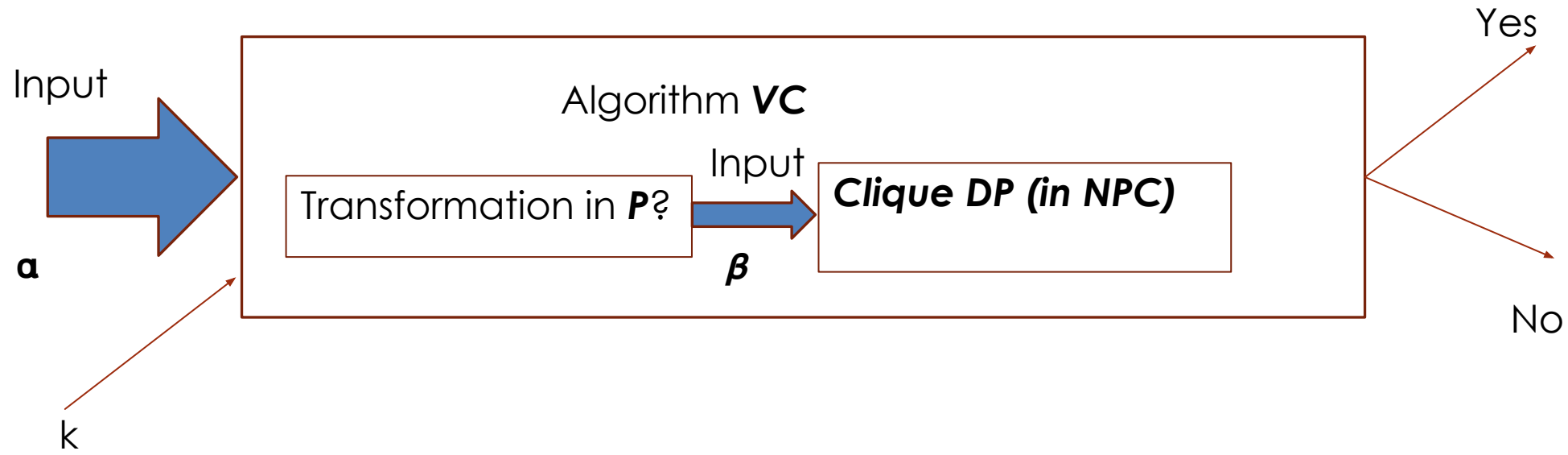
If 3CNF SAT with k clauses is satisfiable, then corresponding graph has clique of size k



Vertex cover (VC) Problem is in NPC



Vertex cover (VC) Problem is in NPC



Vertex cover (VC) is in NPC

VC: Does graph has VC of size k?
Given : CDP is in NPC

Steps:

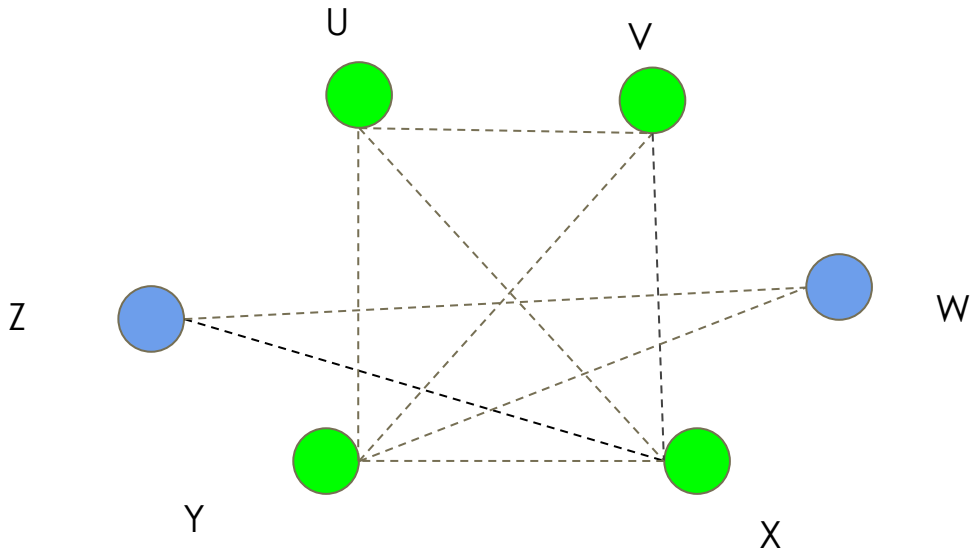
1. Show that VC belongs to NP: Given the certificate, can be verified in polynomial time.
    - Certificate: Set S consisting of k nodes in the VC.
    - Each edges of graph can be checked if it is incident on one of the k vertices O(|E|.k) time
2. Show that VC can be mapped on Clique in polynomial time

Vertex cover (VC) is in NPC

➢ Show that VC can be mapped on Clique in polynomial time

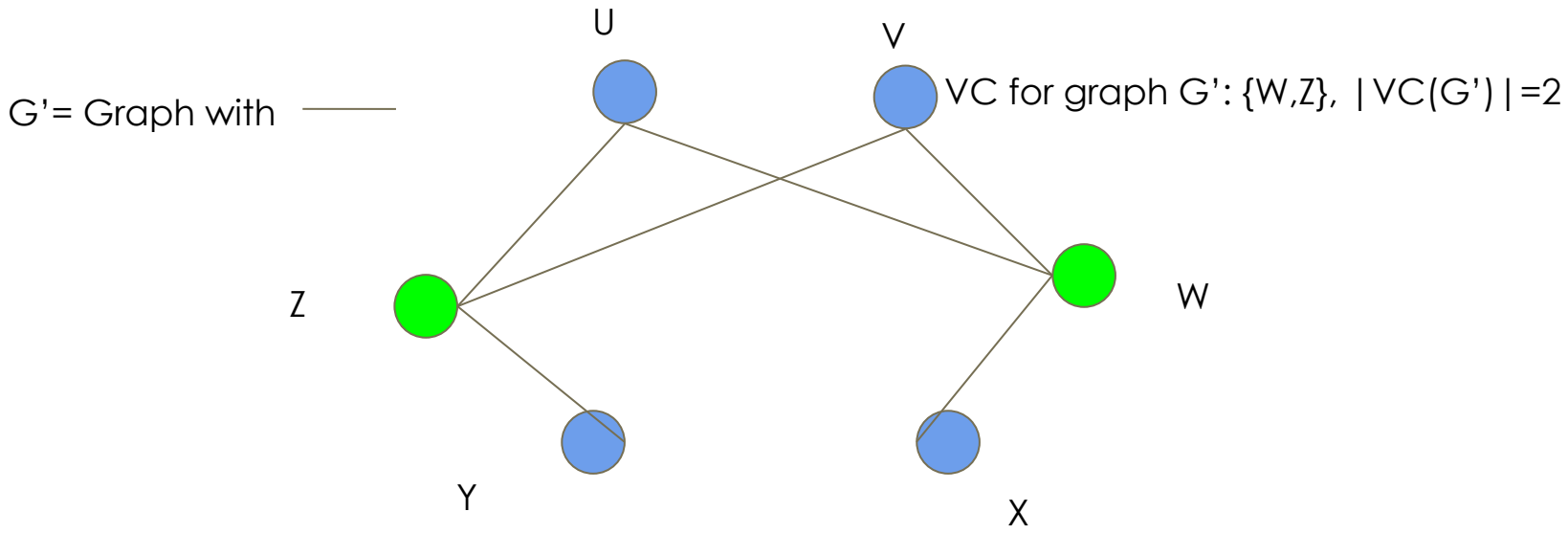
Let G be the given graph

G= Graph with ------- clique for Graph G: {U,V,X,Y}, |CLIQUE(G)|=4



Vertex cover (VC) is in NPC

Let G' be complement of graph G (replace edge by non-edge and vice versa)



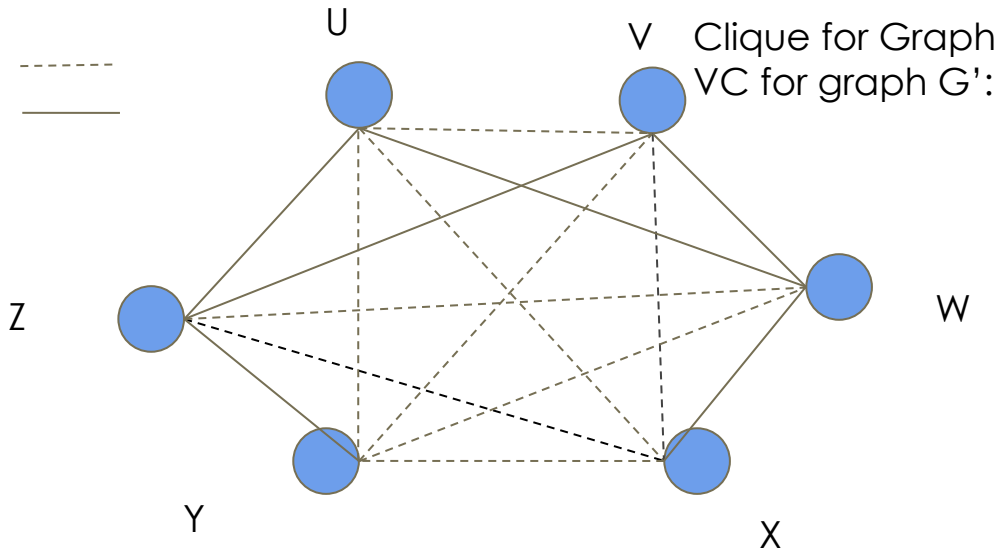
Graph G has clique of size k iff G' has VC of |V|-k

Vertex cover (VC) is in NPC

Let G' be complement of graph G (replace edge by non-edge and vice versa)

G= Graph with ------

G'= Graph with ——



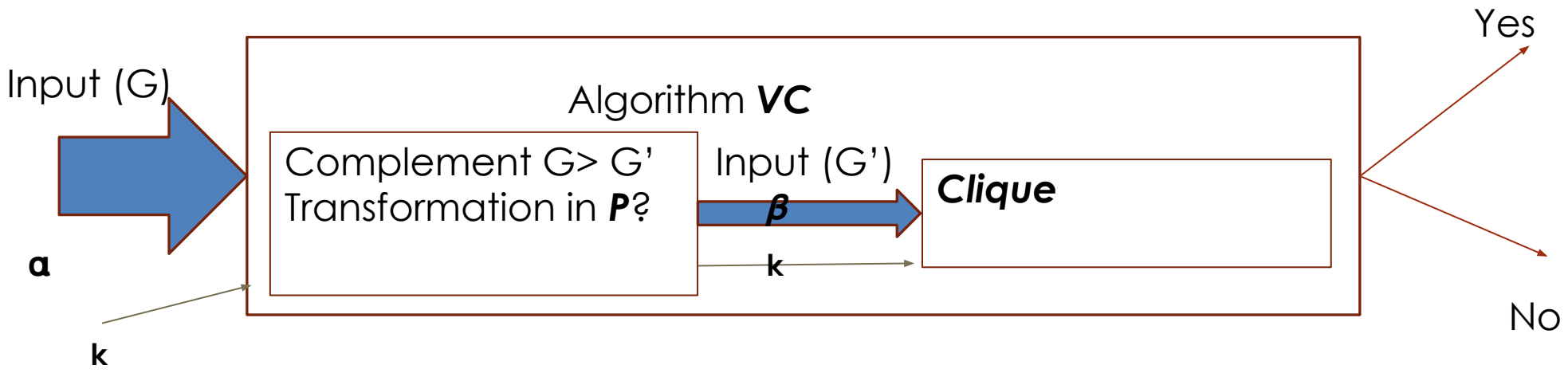
Clique for Graph G: {U,V,X,Y}, |CLIQUE(G)|=4

VC for graph G': {W,Z}, |VC(G')|=2=n-|MAXCLIQUE(G)|

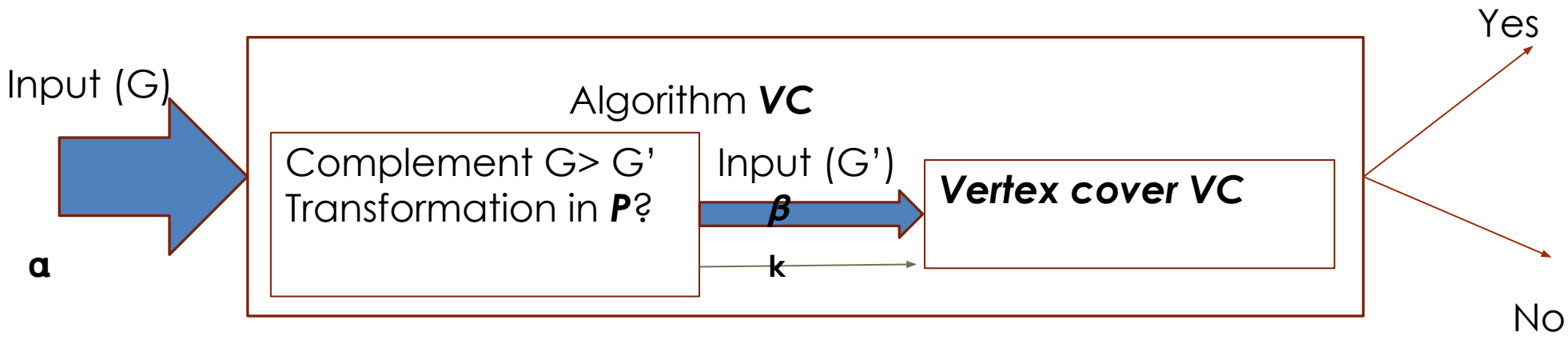
Graph G has clique of size k iff G' has VC of |V|-k

Vertex cover (VC) Problem is in NPC

If graph G has clique of size k iff G' has VC of |V|-k



Hamiltonian Cycle (HC) Problem is in NPC



Hamiltonian Cycle (HC) is in NPC

HC: Does graph has HC?

Certificate: Vertex sequence S forming cycle

Steps:

1. Show that HC belongs to NP:
    - Given the certificate, can be verified in polynomial time.
    - Each pair in the sequence be checked for its existence as an edge.
2. Show that HC can be mapped on VC which in NPC

Approximate Algorithm: Introduction

- Optimality: (Cost C*)
  - Maximization
  - Minimization
- Optimal (C*) Vs Near optimal solution (C)
- Approximation ratio rho (ρ) ≤ 1
  - C/C* or C*/C
  - Algorithm is ρ(n) approximate if it depends on input size n
  - 1 approximate, optimized algorithm (c=c*), if ρ=1
- For many problems - polynomial time approximation algorithm with small constant approximate ratio exists

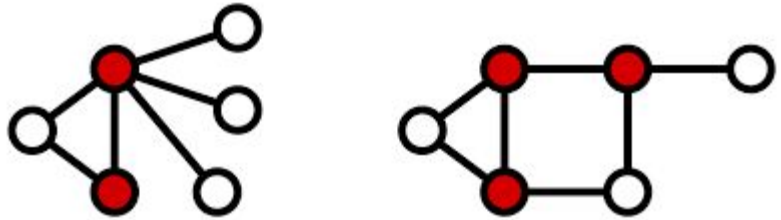
Approximate algorithm

- Approximate scheme which takes instance and value ϵ >0 for fixed ϵ, the scheme is called as (1+ϵ) approximation scheme.
  - $\epsilon \propto$ 1/run time (As ϵ increases running time decreases)
- Polynomial time approximation scheme:
  - If for ϵ>0, the scheme runs in time polynomial in, size n of input.
- Fully Polynomial time approximation scheme:
  - Running time proportional to both 1/ϵ and size n of input instance.

Vertex cover (VC)

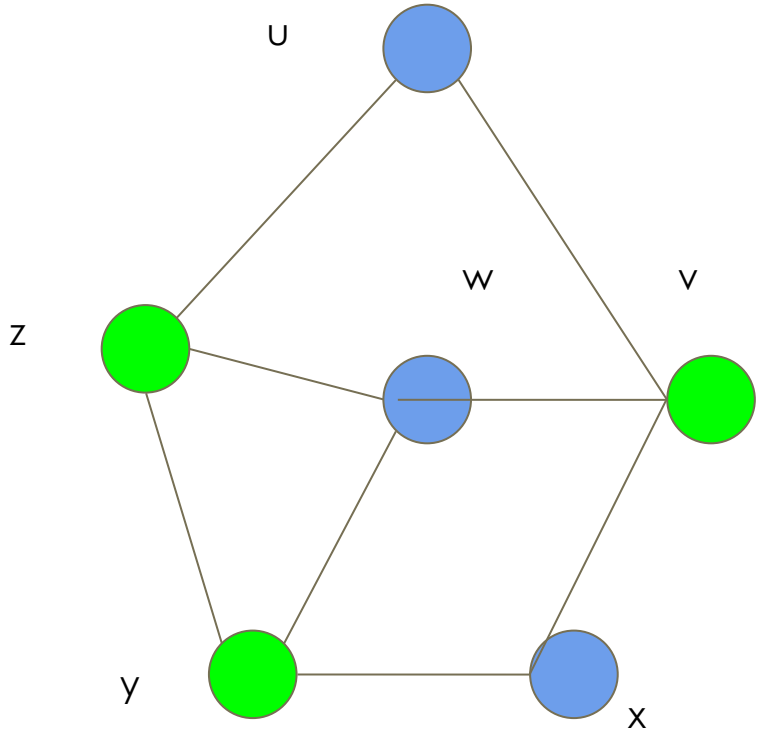
- Subset of vertices which covers (all edges of graph are coincident on at least one of the vertex in the VC) all the edges of graph.

- VC of G(V,E) is subset $V' \subseteq V$ such that if $(u,v) \in E$ then either $u \in V'$ or $v \in V'$.
  - Optimal vertex cover problem is to find vertex cover of minimum size.



Approximate algorithm:Vertex cover (VC)

- Optimal VC: {y,z,v}

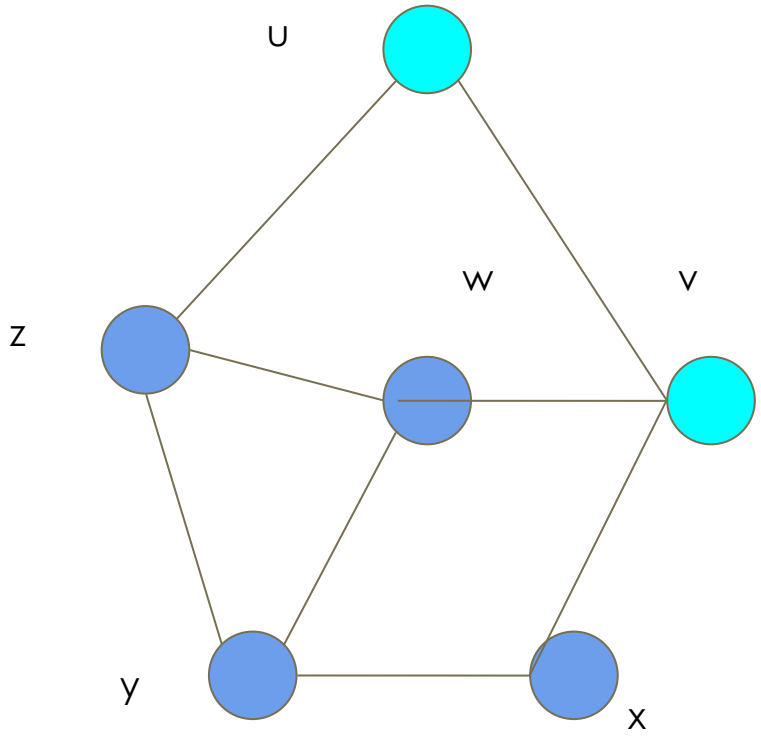


Approximate algorithm:Vertex cover (VC)

Procedure

- Prepare list of edges of a given graph
- Select any edge (u,v),
- include both u and v in approximate vertex cover, delete all edges incident on any of these vertices
- Repeat till all edges are covered

Approximate Vertex cover (VC): Example



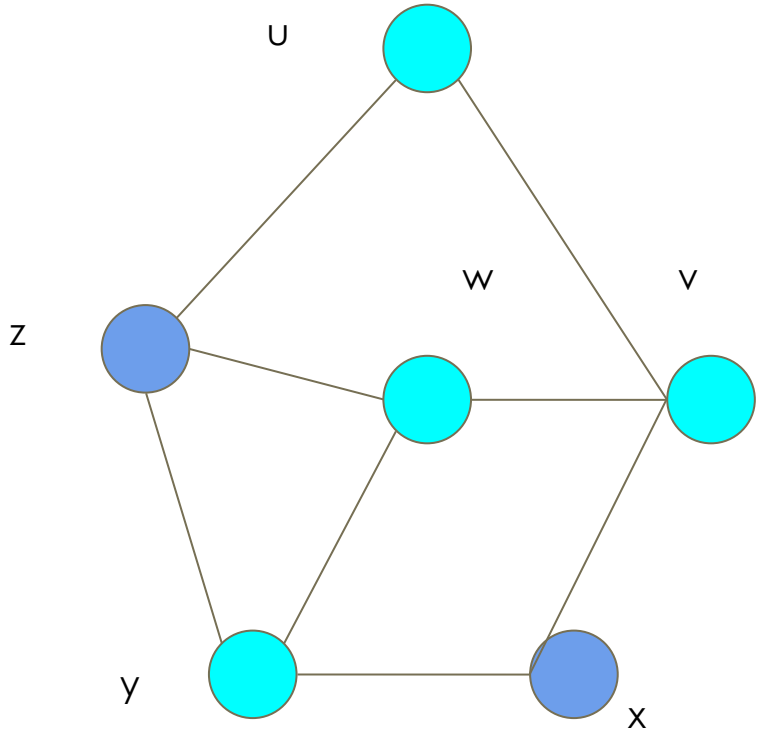
- Prepare list of edges of a given graph {(u,v),(v.x),(x,y),(y,z),(z,u),(y,w),(z,w),(w,v)}
- Select any edge, say (u,v),
- include both u and v in approximate vertex cover, delete all edges incident on these vertices incident on either u or v i.e. (u,v), (z,u), (w,v) and (x,v)

u,v
v,x
x,y
y,z
z,u
y,w
z,w
w,v

Optimal VC: {y,z,v}

Current Approximate VC : { u,v}

Approximate Vertex cover (VC): Example



- Select any other edge, say (y,w),
- include both y and w in approximate vertex cover, delete all edges in the list, incident on these vertices incident on either y or w

(y,w),(y,z),(y,x),(w,z),(w,v)

- Repeat till all edges are deleted

x,y
y,z
y,w
z,w

Optimal VC: {y,z,v}

Current Approximate VC : { u,v,y,w}

Approximate Vertex cover (VC): Analysis

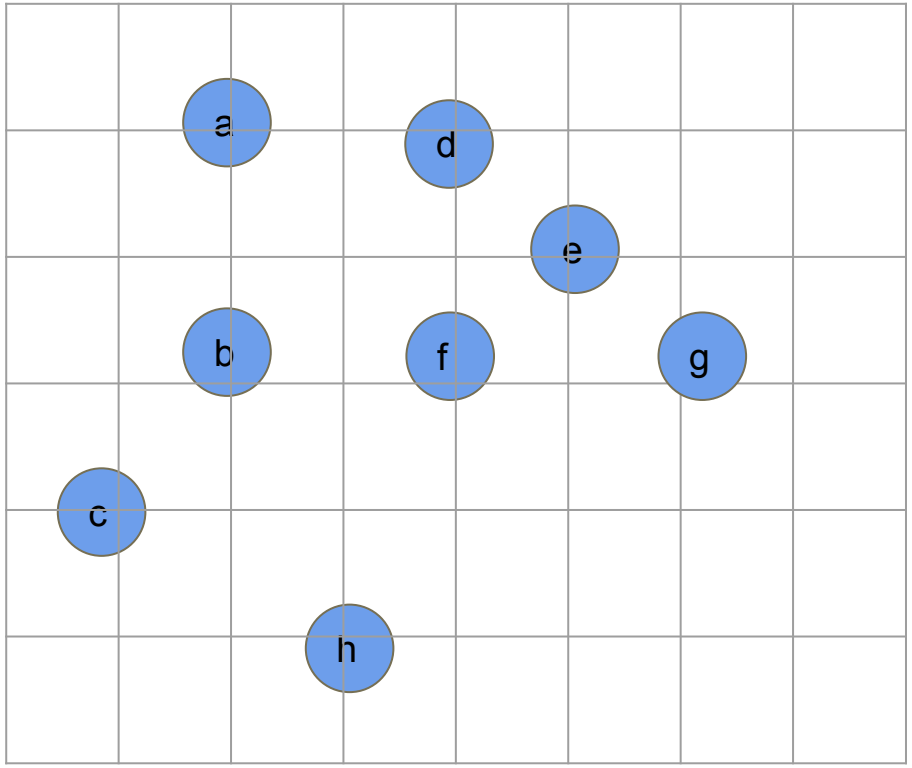
- Copy edge set E in E'
- We select edge (u,v) and put vertices u and v in VC. Delete the edge from E'
- Let A is set of edges selected.
- No two edges in A are covered by same vertex in C
  - Approx cover C=2 | A |
- Let c* be optimal solution (min) $|C^*| \geq |A|, 2|C^*| \geq 2|A|$
- So $C \leq 2|C^*|$

Vertex cover is polynomial time 2 Approximate algorithm

Travelling salesman Problem (TSP)

- Given complete weighted graph G=(V,E) find Hamiltonian Cycle with minimum cost

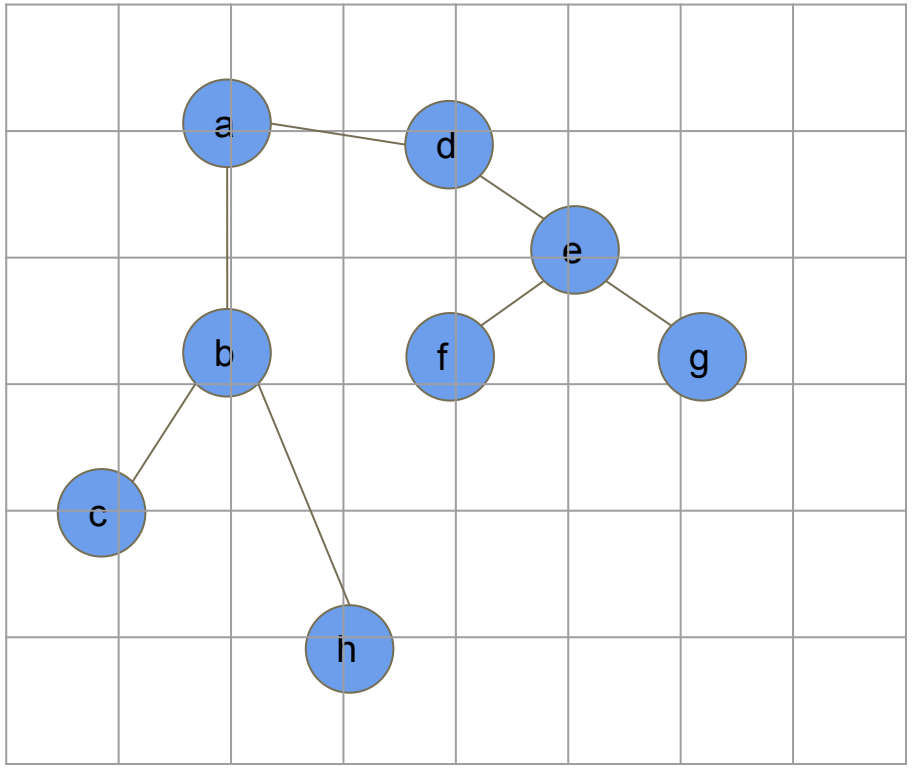
Approximate algorithm:(TSP)



Complete graph is given

1. Compute MST

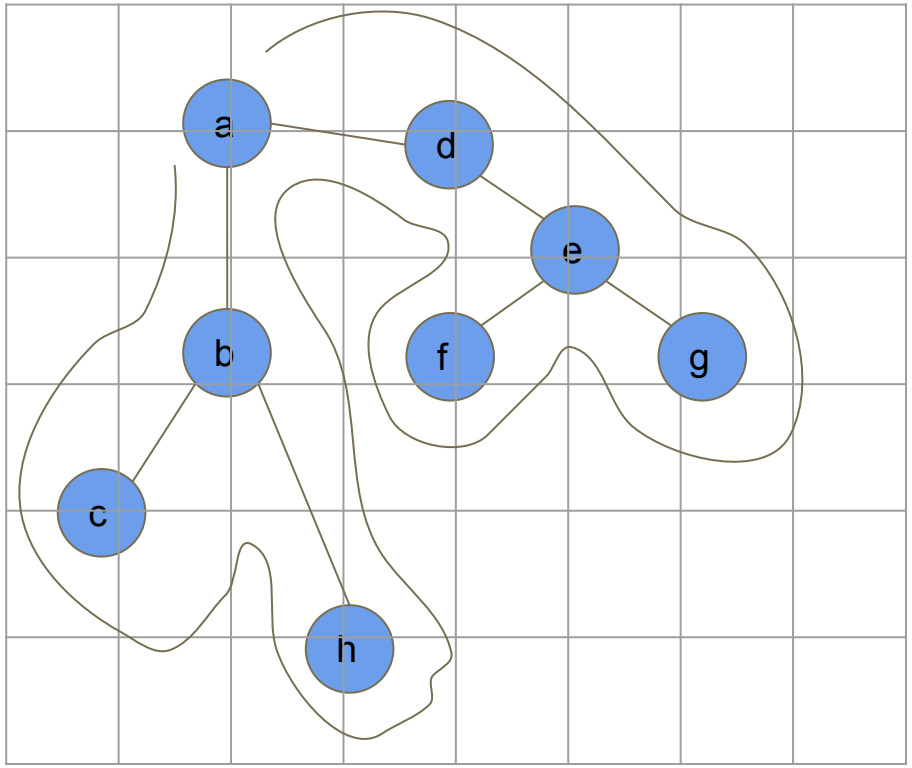
Approximate algorithm:(TSP)



MST starting from 0

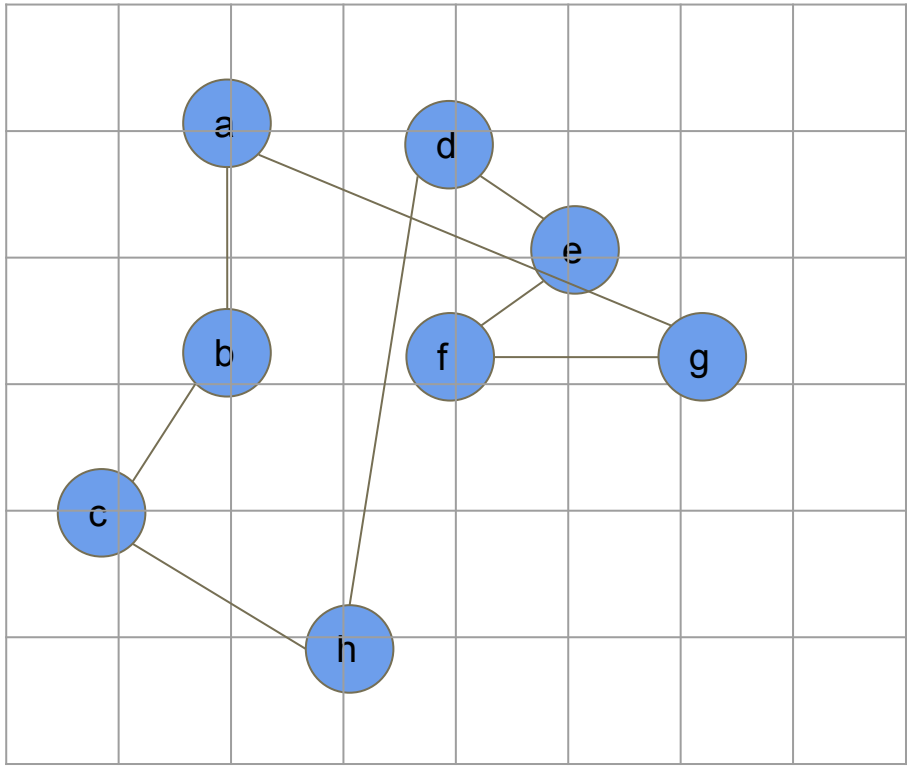
1. Compute MST

Approximate algorithm:(TSP)



- Preorder walk from a : a,b,c,b,h,b,a,d,e,f,e,g

Approximate algorithm:(TSP)

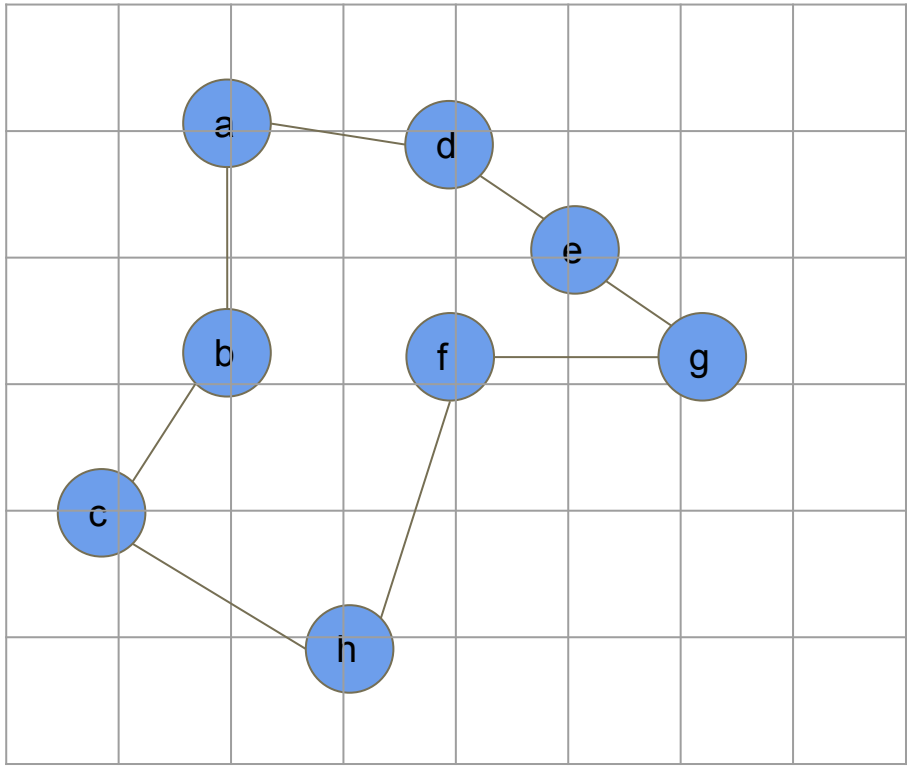


Polynomial time 2 approximate algorithm

- Tour: listing vertices visited first
  a,b,c,b,h,b,a,d,e,f,e,g

  a,b,c,h,d,f,e,g,a
- Approximate tour length ->19
- Optimal Tour length ->14.7

Approximate algorithm:(TSP)



MST starting from a

1. Optimal TSP
   a,b,c,h,f,g,e,d,a

1. Optimal Tour-14.7

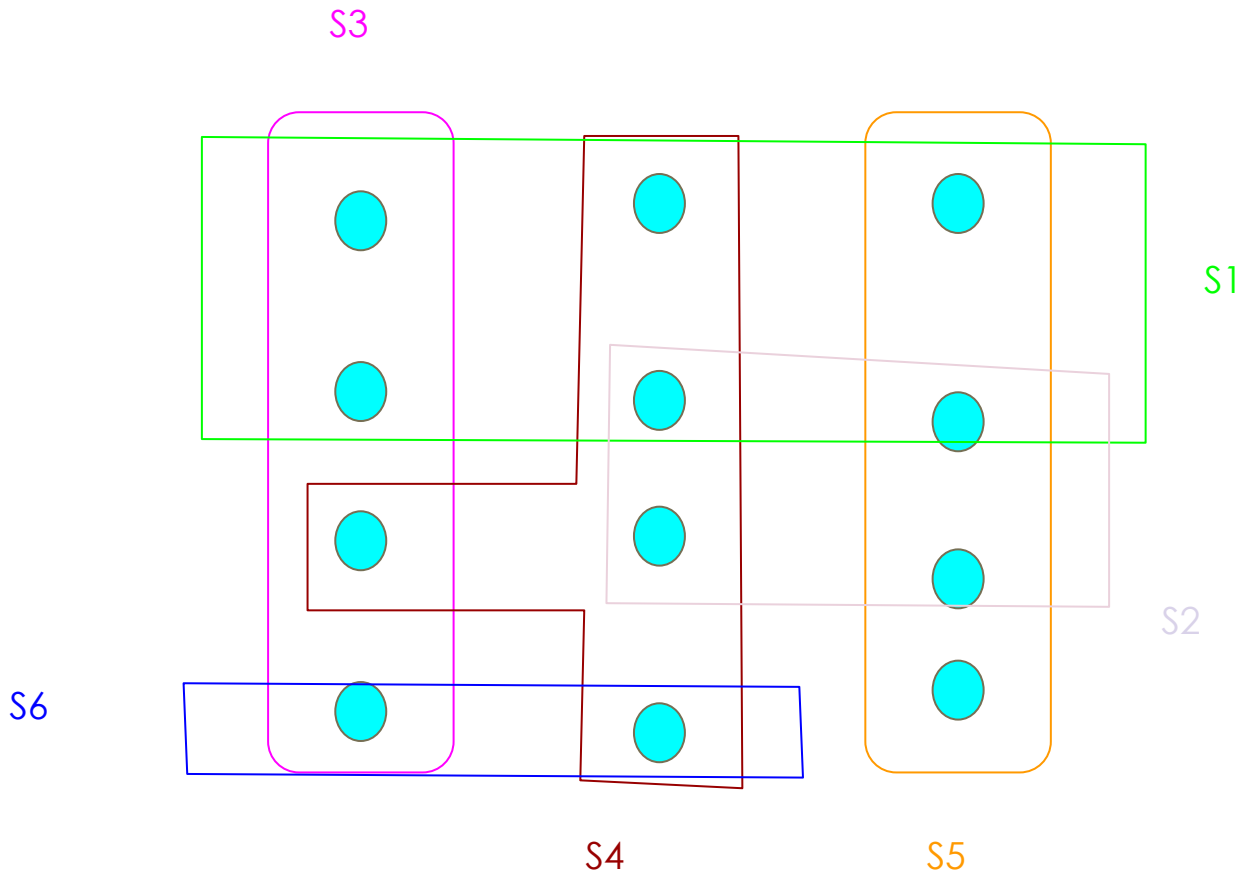
Approx TSP tour: Polynomial time 2 AA

- Let H* be optimum tour
- MST(T) can be obtained by deleting any vertex from this tour
  - c(T)<=c(H*)
- Full walk of T consists of a,b,c,b,h,b,a,d,e,f,e,g,e,d,a
  - Full walk traverses every edge twice
    - c(W)=2c(T)
    - c(W)=2c(H*)
- Approx TSP tour is poly time 2

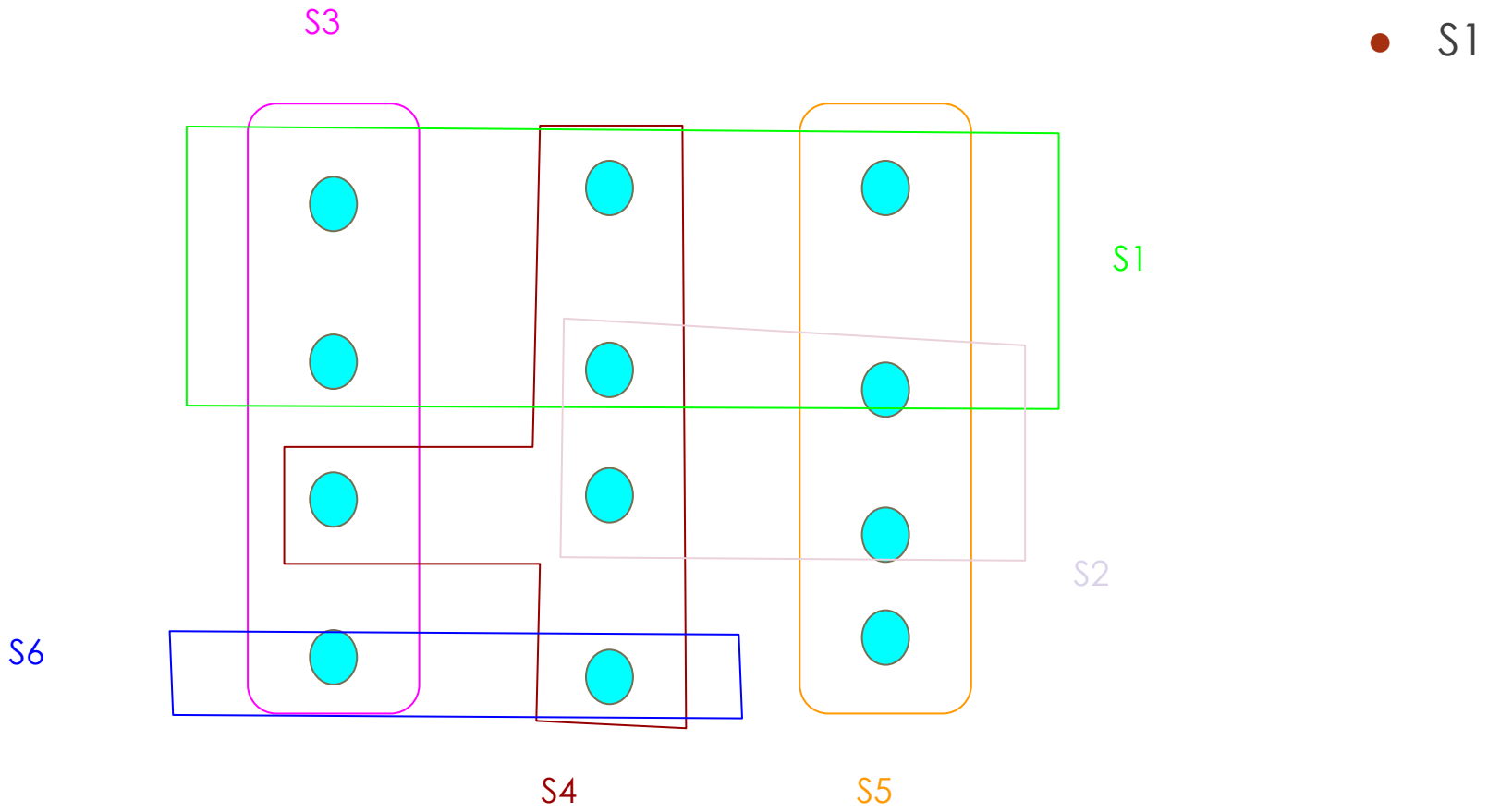
Set covering Problem

- Given sub-sets with different number of elements, find set with minimum number of subset which covers all elements of all subsets.
- Greedy solution: Pick at each stage , set S that covers maximum number of remaining elements that are uncovered. (Polynomial time
(ln |S| +1)approximate algorithm

Set covering Problem



Set covering Problem



Thank you…..