CO4
NP - HARD AND COMPLETE

PREPARED BY

B SREEDHAR

ASSISTANT PROFESSOR, CSE-H, KLEF

BASIC CONCEPT

- The computing times of algorithms fall into two groups. Polynomial and Exponential
- **Group1**– Consists of problems whose solutions are bounded by the polynomial of small degree.
- Example – Binary search O(logn) , sorting O(nlogn), matrix multiplication $O(n^{2.81})$
- **Group2** – Contains problems whose best known algorithms are non polynomial.

  Example –Traveling salesperson problem $0(2^n)$, knapsack problem $0(2^{n/2})$ etc.

NP-HARD AND NP-COMPLETE

There are two classes of non polynomial time problems

1. **NP-Complete** - Have the property that it can be solved in polynomial time if all other NP-Complete problems can be solved in polynomial time.

2. **NP-Hard** - If it can be solved in polynomial time then all NP-Complete can be solved in polynomial time.

- "All NP-Complete problems are NP-Hard but not all NP Hard problems are not NP-Complete"

NP-HARD AND NP-COMPLETE

Deterministic and Non-Deterministic Algorithms

Algorithms with the property that the result of every operation is uniquely defined are termed deterministic. Such algorithms agree with the way programs are executed on a computer.

When the outcome is not uniquely defined but is limited to a specific set of possibilities, we call it non deterministic algorithm. To specify such algorithms in SPARKS, we introduce three statements

i) choice(S) : arbitrarily chooses one of the elements of the set S.

ii) failure : Signals an unsuccessful completion.

iii) Success : Signals a successful completion.

NP-HARD AND NP-COMPLETE

Deterministic Example

```
Algorithm Lsearch (A, n, x)  {
   for i:= 1 to n do  {
            if(a[i]=x) then
                        return i;
   }
   return 0;
}
```

NP-HARD AND NP-COMPLETE

Non-Deterministic Example

```
Algorithm Search(A, n, x)  {
   j:= choice(1,n);
   if(a[j]=x) then   {
            return j;
            success();
   }
   return 0;
   failure();
}
```

Note: The statements choice(), success(), and Failure() are non-deterministic. These statements are required 1 unit of time, so the time taken to execute the algorithm is O(1)

NP-HARD AND NP-COMPLETE

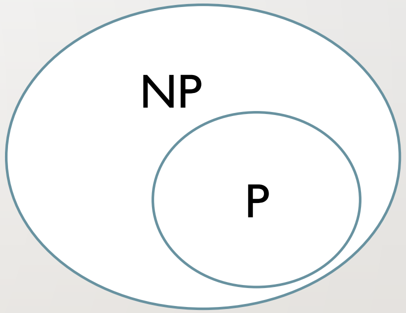
Two Classes of Algorithms

P – set of those deterministic algorithms which are taken polynomial time. Ex: Linear search, binary search, bubble sort, merge sort, single source shortest path, minimum cost spanning tree, Huffman coding, etc.,

NP – These algorithms are non-deterministic but they take polynomial time.

NP-HARD AND NP-COMPLETE

- Here, we are trying to convert all the exponential time algorithm into non-deterministic with polynomial time. Consider all the problems of exponential and make to find the relation by using **Satisfiability**.



NP-HARD AND NP-COMPLETE

SATISFIABILITY

- Let x_1 ,x_2 ,x_3….,x_n denotes Boolean variables.
- Let $\bar{x}_i$ denotes the negation of xi .
- A literal is either a variable or its negation.
- A formula in the prepositional calculus is an expression that can be constructed using literals and the operators ∧(AND) and V(OR).
- A formula is in Conjunctive Normal Form (CNF) iff it is represented as $\wedge C_i$. Disjunctive Normal Form (DNF) represented as $\vee C_i$.

NP-HARD AND NP-COMPLETE
SATISFIABILITY

- The satisfiability problem is to determine if a formula is true for some assignment of truth values to the variables
- CNF-Satisability is the satisfiability problem for CNF formulas.

- For **Example**: Consider the Boolean variable $X_i = \{ X_1, X_2, X_3 \}$

$$CNF = (X_1 \cup \sim X_2 \cup X_3) \cap (\sim X_1 \cup X_2 \cup \sim X_3)$$

Now, identify the possible values of the X_i →

X_1	X_2	X_3
0	0	0
0	0	1
.	.	.
.	.	.
1	1	1

Total 8 possible values, to compute these values it require 2^n unit of time

NP-HARD AND NP-COMPLETE

SATISFIABILITY

- Substitute these values into our CNF equation and check whether possibility is true.

CNF = $(X_1 \cup \sim X_2 \cup X_3) \cap (\sim X_1 \cup X_2 \cup \sim X_3)$

Lets $X_1 = 0 \quad \sim X_1 = 1$

$X_2 = 0 \quad \sim X_2 = 1$

$X_3 = 1 \quad \sim X_3 = 0$

CNF = $(0 \cup 1 \cup 1) \cap (1 \cup 0 \cup 0)$

CNF = $(1) \cap (1)$

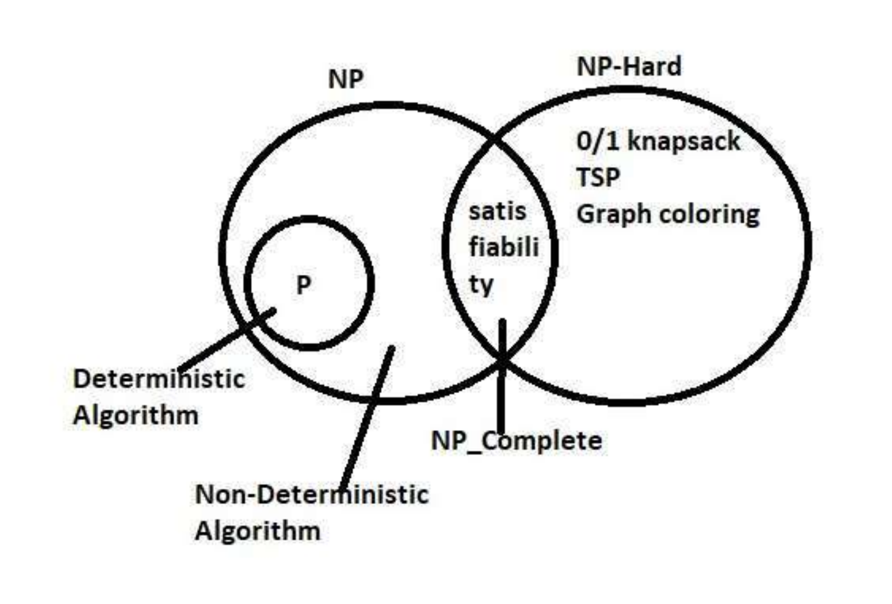
CNF = TRUE

- If satisfiability solve in polynomial time and all the exponential algorithms can solve in polynomial time.

NP-HARD AND NP-COMPLETE
REDUCTION

Let L1 and L2 be problems. L1 reduces to L2 (L1 α L2) if and only if there is a deterministic polynomial time algorithm to solve L1 that solves L2 in polynomial time.

➢ If L1 α L2 and L2 α L3 then L1 α L3.

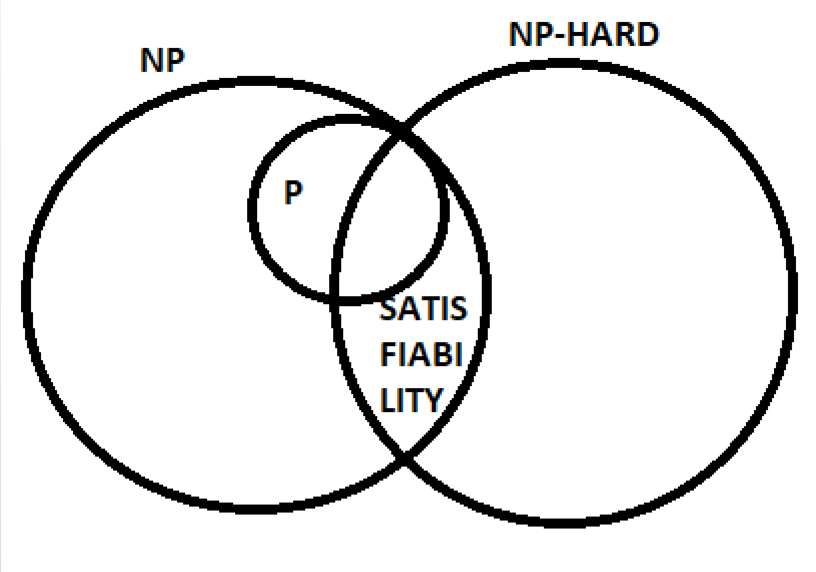


- Satisfiability α 0/1knapsack
- If the problem 0/1 knapsack is solved in polynomial time, then the same will solve by the satisfiability.
- If satisfiability have non-deterministic polynomial time algorithm, then that problem is NP-Complete.

- If we are able to prove that P=NP, then non-deterministic will be convert into deterministic, that is P ⊆ NP. This can be done by using cook's theorem

NP-HARD AND NP-COMPLETE
COOK'S THEOREM

- Satisfiability is in P if and only if P = NP. Means that if satisfiability is deterministic then P=NP.
- NP-Hard - A problem L is NP-hard if any only if satisfiability reduces to L.
- NP-complete - A problem L is NP-complete if and only if L is NP-hard and L ϵ NP.

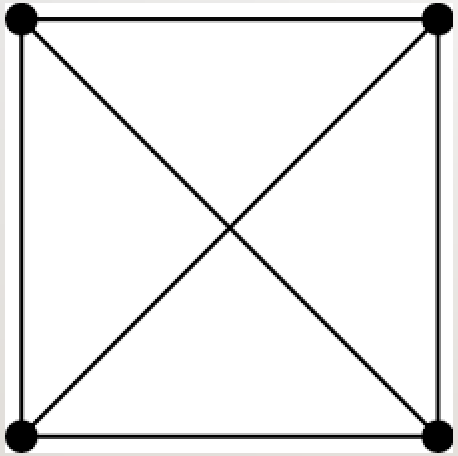


NP-HARD AND NP-COMPLETE

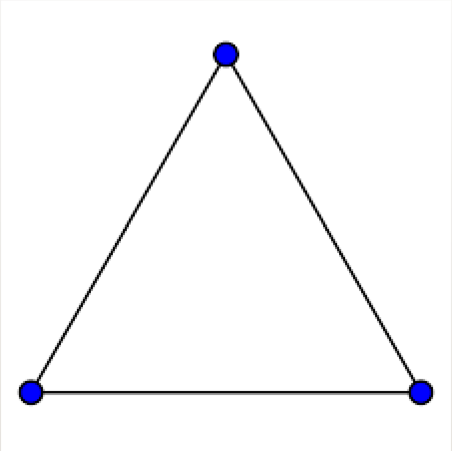
NP-Hard	NP-Complete
NP-Hard problems(say X) can be solved if and only if there is a NP-Complete problem(say Y) that can be reducible into X in polynomial time.	NP-Complete problems can be solved by a non-deterministic Algorithm/Turing Machine in polynomial time.
To solve this problem, it do not have to be in NP .	To solve this problem, it must be both NP and NP-hard problems.
Do not have to be a Decision problem.	It is exclusively a Decision problem.
Example: Halting problem, Vertex cover problem, etc.	**Example**: Determine whether a graph has a Hamiltonian cycle, Determine whether a Boolean formula is satisfiable or not, Circuit-satisfiability problem, etc.

NP-HARD GRAPH PROBLEMS
CLIQUE DECISION PROBLEM (CDP)

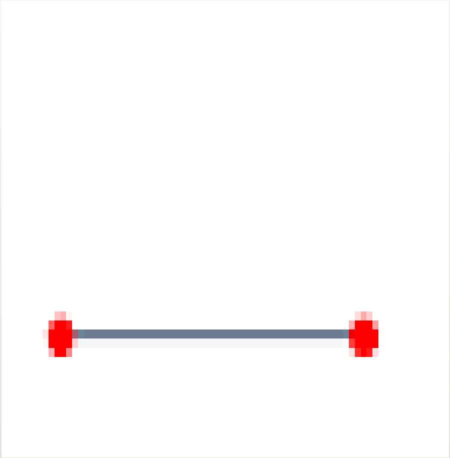
- CDP is NP-Hard problem.
- Clique is a maximal complete sub graph of a graph G = (V,E)
    - Size of a clique is the number of vertices in it



Complete graph of 4 vertices



Complete graph of 3 vertices

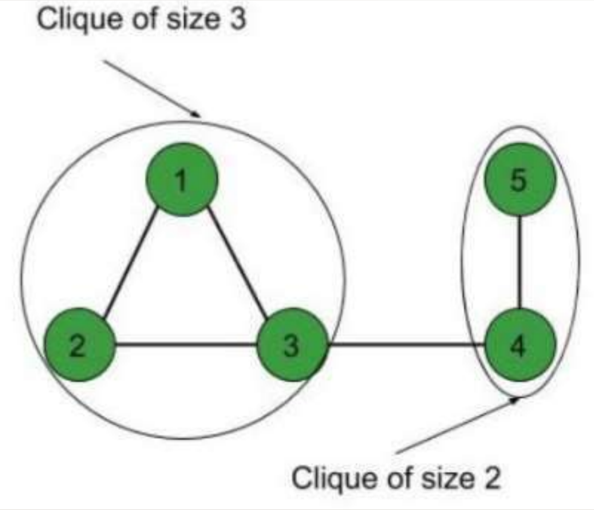


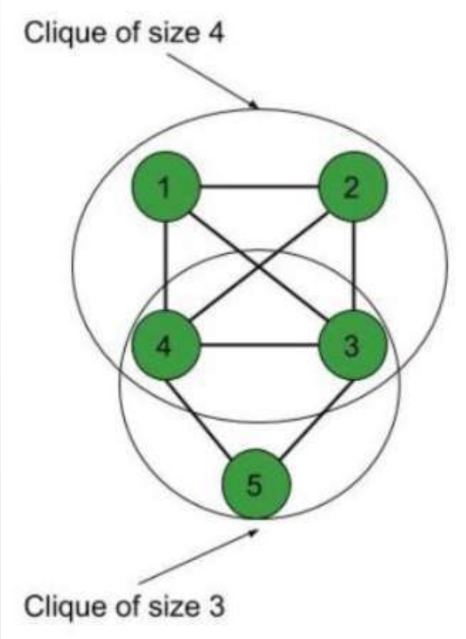
Complete graph of 2 vertices

Property of
Complete graph $|V| = n$
$|E| = n(n-1)/2$

NP-HARD GRAPH PROBLEMS
CLIQUE DECISION PROBLEM (CDP)

- CDP is NP-Hard problem.
- Clique is a maximal complete sub graph of a graph G = (V,E)
    - Size of a clique is the number of vertices in it





- It is not complete graph.
- The sub graph which contain 4 vertices is complete graph, so, the sub graph here called as Clique of size 4

NP-HARD GRAPH PROBLEMS
PROVE CDP IS NP-HARD

The general method for prove the problem is NP-Hard

- Let consider our problem is L_2, we have to prove that L_2 is NP-Hard.
- For proving, we can select a problem a problem L_1 is already as NP-Hard, and we have to show that $L_1 \alpha L_2$.
- Now take an example instance i_1 if L_1 and prepare an example instance i_2 of L_2, such that the example instance i_2 can solve in polynomial time, then the example instance i_1 is also solve in polynomial time

- In our problem we have to prove that CDP is NP-Hard, that is

Satisfiability α CDP

Here, Satisfiability is NP-Hard, which is already known, now we have to prove that CDP is NP-Hard.

NP-HARD GRAPH PROBLEMS
PROVE CDP IS NP-HARD

Satisfiability α CDP

$F = (x_1 \cup x_2) \cap (\sim x_1 \cup \sim x_2) \cap (x_1 \cup x_3)$

C_1 C_2 C_3

From this formula we have to prepare a graph, such that which is having a clique and that clique should be of size k(where k is number of clauses c1, c2, c3)

- In a graph G, $V = \{ <a.i> \mid a \in C_i \}$ $E = \{ <a. i>, <b, j> \mid i \neq j \text{ and } b \neq \sim a \}$
- Here, same vertex in the same clause cannot from as an edge
- Cannot connect the edge with the negation.

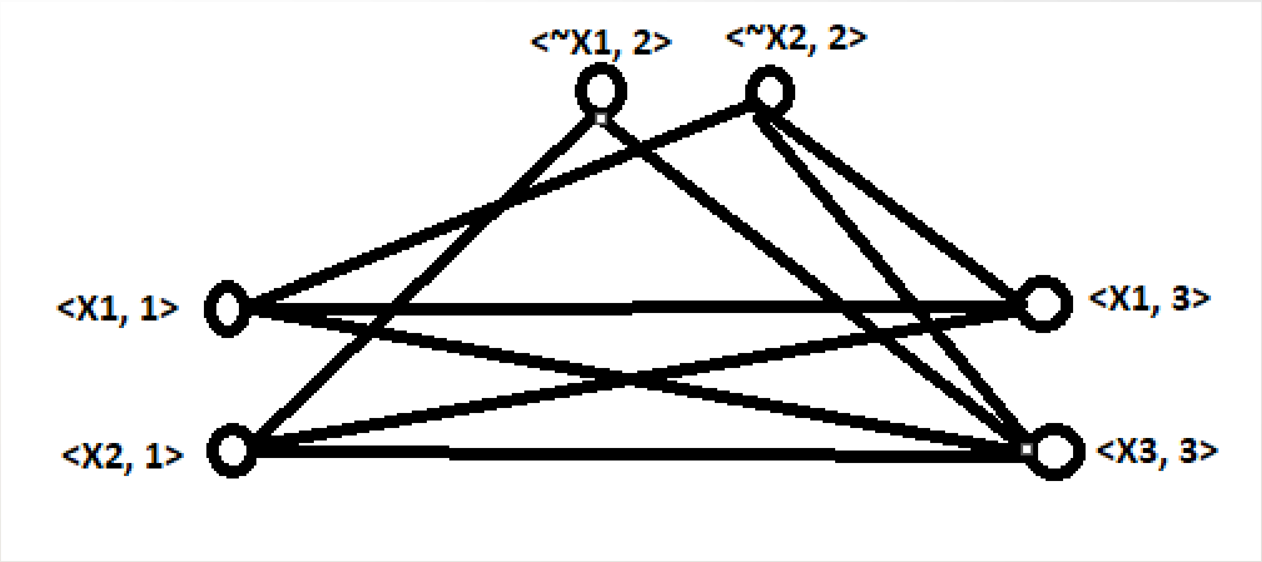
NP-HARD GRAPH PROBLEMS

PROVE CDP IS NP-HARD

The graph G solves in polynomial Time so, Formula F is also solves in polynomial time.
From the graph K=3, Clique of size = 3
Because the formula having 3 clauses (C1,C2,C3) and if the size with clique 3 solves, the F is also solves
How ? Means

X1 X2 X3
0 1 1 select the vertex from the graph

Substitute these values into the F.
F = 1 (True)
Take any clique of size 3, it becomes true. Finally, we say that F is NP-Hard and G is also NP- Hard



There fore CDP is NP-Hard

NP-HARD GRAPH PROBLEMS
NODE COVER DECIAION PROBLEM (NCDP)

Def:

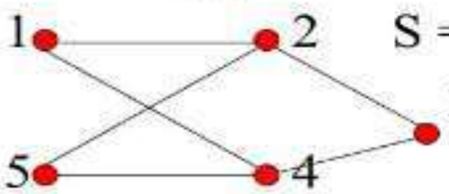
A set S ⊆ V is a node cover for graph G (V, E) if and only if all edges in E are incident on at least one vertex S.

The size |S| is the number of vertices in S.

For Example:

A set S ∈ V is a node cover for a graph G(V,E) if and only if all edges in E are incident to at least one vertex in S.

Size of |S| of the cover is the number of vertices in S



S = {2,4} is anode cover of size 2

S = {1,3,5} size 3

In NCDP , we are given a graph and integer K, determine whether G has a node cover of size K

NP-HARD GRAPH PROBLEMS
PROVE NCDP IS NP-HARD

We have to prove that

Clique Decision Problem ∝ Node cover decision problem

Solution:

Let G = (V, E) and K define an instance of clique decision problem.

We construct a graph G' such that G' has a node cover of size at most n-k if and only if G has a clique of size at least k.

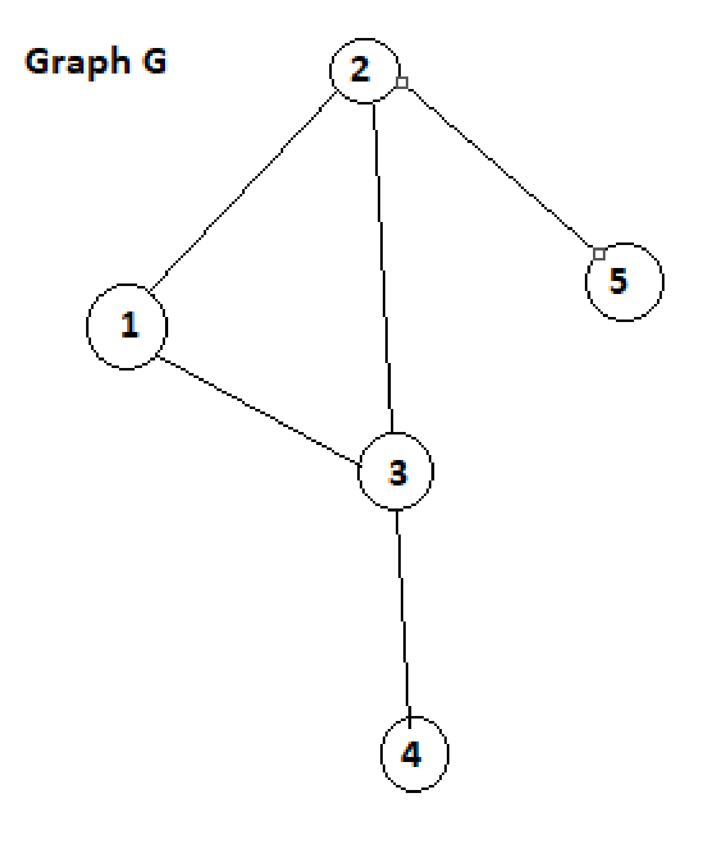
Graph $G^{|} = (V, E^{|})$ where $E^{|} = \{(u, v) \mid u \in v, v \in V \text{ and } (u, v) \notin E\}$

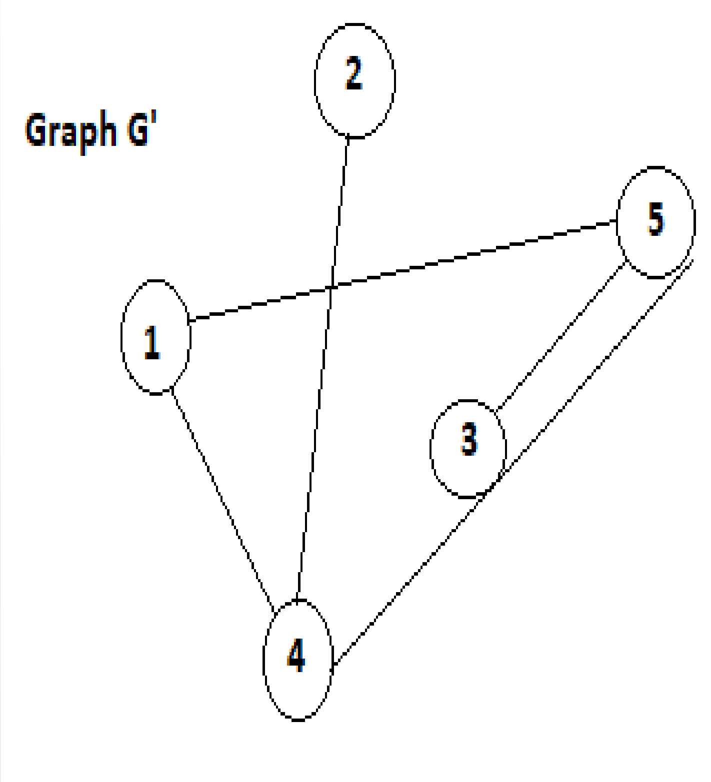
Let K be aby clique in G, since there are no edges in ~E connecting vertices in K, the remaining n-|K| vertices in G' must covers all edges in E'. (E' → edges are available in G, but not G')

Similarly, if S is a node cover of G', then V-S must form a complete sub graph in G.

Since G' can be obtained from G in polynomial time, CDP can be solve in polynomial deterministic time, If we have a polynomial time deterministic algorithm for NCDP.

NP-HARD GRAPH PROBLEMS
PROVE NCDP IS NP-HARD





Graph G' has node cover of {4, 5}
Since every edge of G' is incident either on 4 or 5.
Thus G has a clique n - |k|

$5 - 2 = 3$

$= \{1, 2, 3\}$

Note: Since Satisfiability $\propto$ CDP

CDP $\propto$ NCDP

Because of transitive property

Satisfiability $\propto$ NCDP

Thus NCDP is NP-Hard

AND OR GRAPH DECISION PROBLEM (AOG)

- Many complex problems can be broken down into a series of subproblems such that the solution of all or some of these results in the solution of the original problem.
- These subproblems can be broken down further into sub subproblems, and so on, until the only problems remaining are sufficiently primitive as to be trivially solvable.

AND OR GRAPH DECISION PROBLEM (AOG)

- This breaking down of a complex problem into several subproblems can be represented by a directed graph like structure in which nodes represent problems and descendents of nodes represent the subproblems associated with them

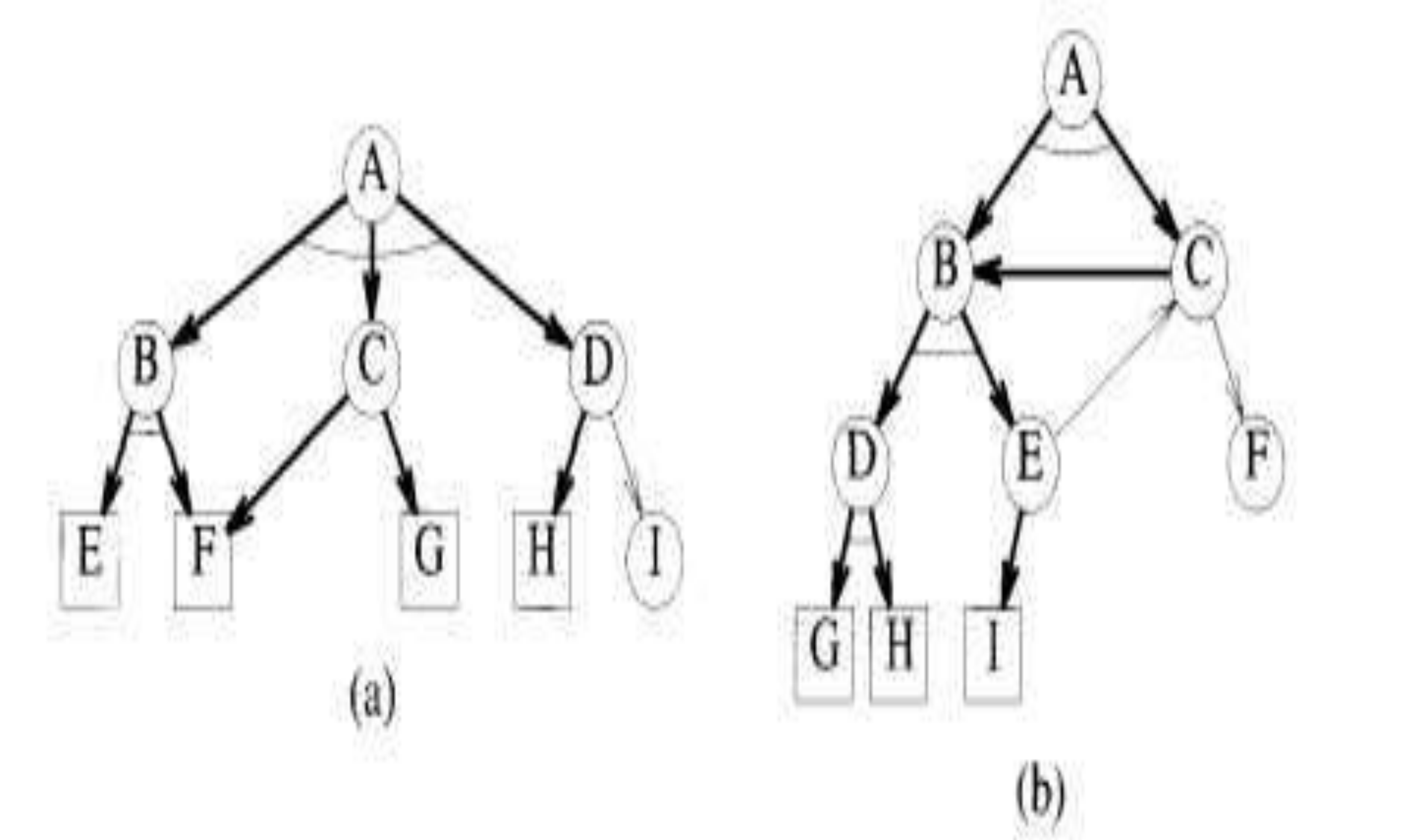
AND OR GRAPH DECISION PROBLEM (AOG)

- To solve OR node requires either all descendents to be solved or only one descendent to be solved.
- To solve AND node requires all the descendents to be solved.
- The AND nodes are drawn with an arc across all edges leaving the node.
- Nodes with no descendents are called terminal. Terminal nodes represent primitive problems and are marked either solvable or not solvable.

AND OR GRAPH DECISION PROBLEM (AOG)

- Solvable terminal nodes are represented by rectangles.
- An AND/OR graph need not always be a tree
- A solution graph is a subgraph of solvable nodes that shows that the problem is solved.

AND OR GRAPH DECISION PROBLEM (AOG)

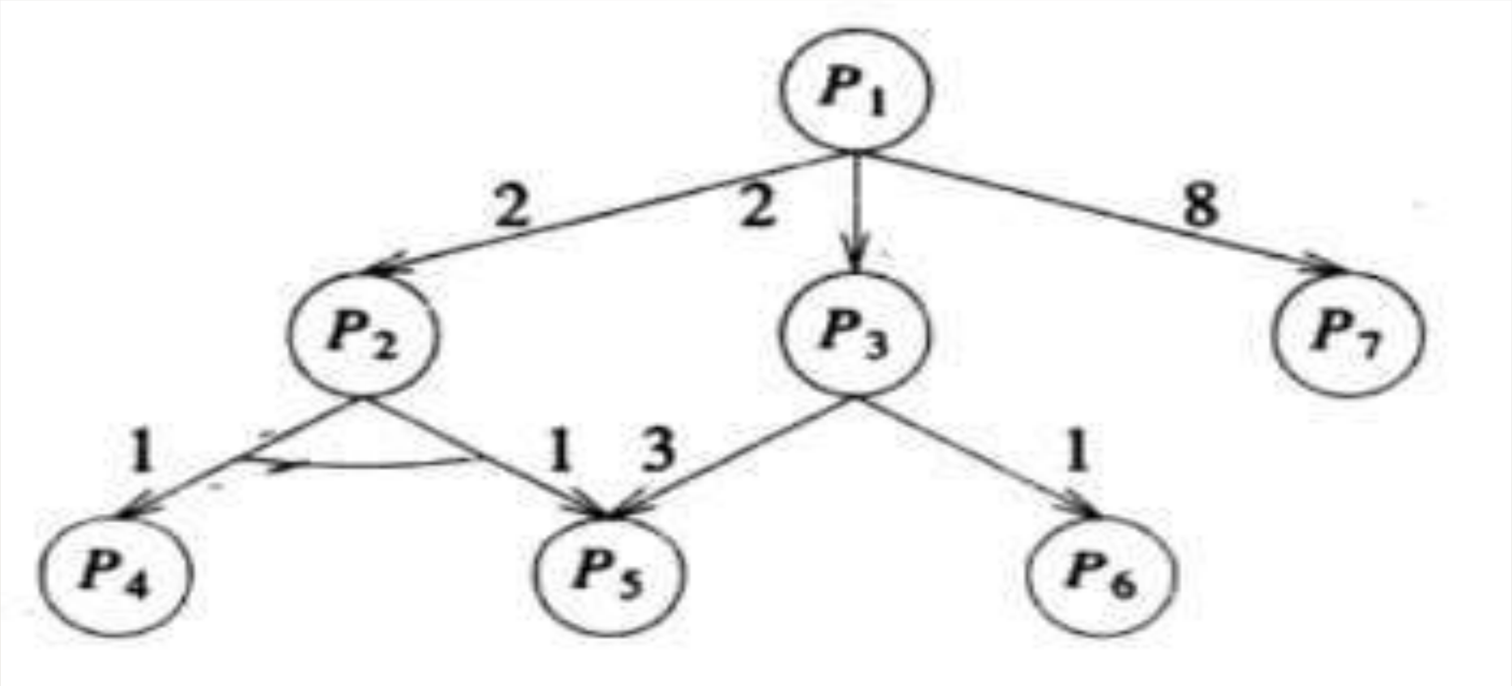


AND OR GRAPH DECISION PROBLEM (AOG)

- Let us assume that there is a cost associated with each edge in the AND/OR graph. The cost of a solution graph H of an AND/OR graph G is the sum of the costs of the edges in H. The AND/OR graph decision problem (AOG) is to determine whether G has a solution graph of cost at most k, for k a given input.

AND OR GRAPH DECISION PROBLEM (AOG)

- Example: Consider the directed graph of the below figure(AOG). The problem to be solved is P1.



AND OR GRAPH DECISION PROBLEM (AOG)

- To do this, one can solve node P2, P3, or P7 as P1 is an OR node.
- The cost incurred is then either 2, 2, or 8 (i.e., cost in addition to that of solving one of P2, P3, or P7).
- To solve P2, both P4 and P5 have to be solved, as P2 is an AND node. The total cost to do this is 2.

AND OR GRAPH DECISION PROBLEM (AOG)

- To solve P3, we can solve either P5 or P6.
- The minimum cost to do this is 1.
- Node P7 is free.
- In this example, then, the optimal way to solve P1 is to solve P6 first, then P3, and finally P1.
- The total cost for this solution is 3

AND OR GRAPH DECISION PROBLEM (AOG)

Theorem 11.7 CNF-satisfiability $\propto$ the AND/OR graph decision problem.

Proof: Let P be a propositional formula in CNF. We show how to transform a formula P in CNF into an AND/OR graph such that the AND/OR graph so obtained has a certain minimum cost solution if and only if P is satisfiable. Let

$$P = \bigwedge_{i=1}^{k} C_i, \quad C_i = \bigvee l_j$$

where the l_j's are literals. The variables of P, $V(P)$ are $x_1, x_2, \ldots, x_n$. The AND/OR graph will have nodes as follows:

1. There is a special node S with no incoming arcs. This node represents the problem to be solved.

AND OR GRAPH DECISION PROBLEM (AOG)

2. The node S is an AND node with descendent nodes $P, x_1, x_2, \ldots, x_n$.

3. Each node x_i represents the corresponding variable x_i in the formula P. Each x_i is an OR node with two descendents denoted Tx_i and Fx_i respectively. If Tx_i is solved, then this will correspond to assigning a truth value of true to the variable x_i. Solving node Fx_i will correspond to assigning a truth value of false to x_i.

4. The node P represents the formula P and is an AND node. It has k descendents $C_1, C_2, \ldots, C_k$. Node C_i corresponds to the clause C_i in the formula P. The nodes C_i are OR nodes.

5. Each node of type Tx_i or Fx_i has exactly one descendent node that is terminal (i.e., has no edges leaving it). These terminal nodes are denoted $v_1, v_2, \ldots, v_{2n}$.

To complete the construction of the AND/OR graph, the following edges and costs are added:

1. From each node C_i an edge $\langle C_i, Tx_j \rangle$ is added if x_j occurs in clause C_i. An edge $\langle C_i, Fx_j \rangle$ is added if $\bar{x}_j$ occurs in clause C_i. This is done for all variables x_j appearing in the clause C_i. Clause C_i is designated an OR node.

2. Edges from nodes of type Tx_i or Fx_i to their respective terminal nodes are assigned a weight, or cost of 1.

3. All other edges have a cost of 0.

AND OR GRAPH DECISION PROBLEM (AOG)

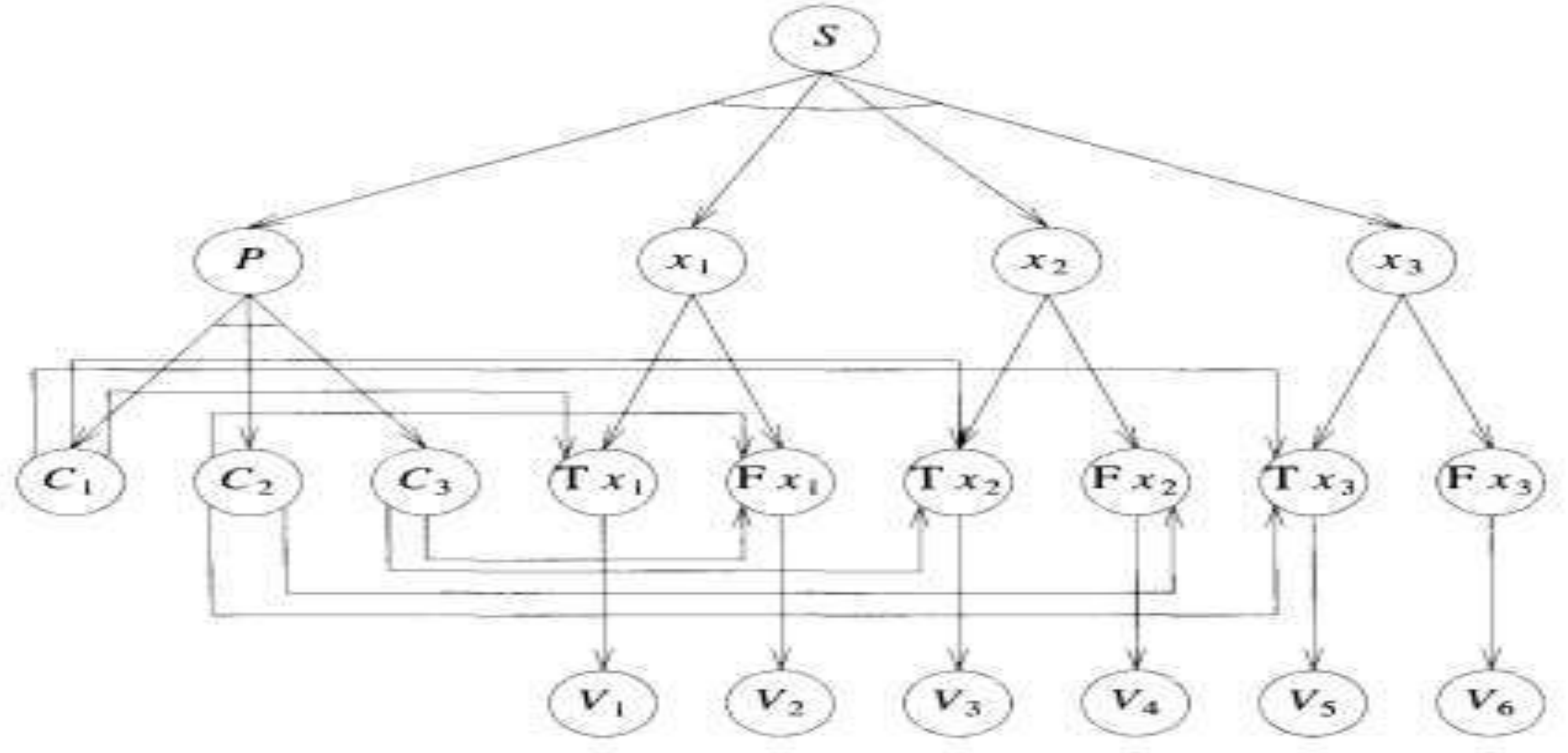
Example 11.18 Consider the formula

$$P = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2); \quad V(P) = x_1, x_2, x_3; \quad n = 3$$

Figure 11.16 shows the AND/OR graph obtained by applying the construction of Theorem 11.7.

The nodes Tx_1, Tx_2, and Tx_3 can be solved at a total cost of 3. The node P costs nothing extra. The node S can then be solved by solving all its descendent nodes and the nodes Tx_1, Tx_2, and Tx_3. The total cost for this solution is 3 (which is n). Assigning the truth value of true to the variables of P results in P's being true. □

AND OR GRAPH DECISION PROBLEM (AOG)



AND nodes joined by arc
All other nodes are OR

Questions:

1. Explain the terminology used in AOG?
2. Prove that CNF-Satisfiability can be reduced to AOG decision problem

THANK YOU