**SURVEY ON NP – COMPLETE PROBLEMS**

by

D. Shashank - 13BCE1036

Harshwardhan - 13BCE1052

S. Nikesh Kumar Reddy - 13BCE1136

*A project report submitted to the*

**SCHOOL OF COMPUTING SCIENCES ENGINEERING**

*In partial fulfillment of the course CSE202 for degree*

*of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE ENGINEERING**

**VIT UNIVERSITY, CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600 127.**

**MAY 2015**

**ACKNOWLEDGEMENT**

**Course Code: CSE202**

**Course Title: Algorithm Analysis and Design**

This project report entitled “**SURVEY ON NP – COMPLETE PROBLEMS”** is a bonafide work of **SHASHANK. D (13BCE1036), HARSH (13BCE1052) and NIKESH. S (13BCE1136)** carried out the Project work under the supervision and guidance of **Prof. JEGANNATHAN. L.** We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof.** **JEGANNATHAN. L,** Associate Professor, School of Computing Sciences and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work

**TABLE OF CONTENTS**

1. Abstract
2. Sub – Graph Isomorphism
3. Knapsack Problem
4. Hamiltonian Cycle
5. Travelling Salesman Problem
6. Quadratic Assignment Problem
7. Sub-set Sum Problem

**ABSTRACT**

All the decidable problems which are in existence can be classified into three classes based on the level of complexity namely P, NP and NP-Complete.

P: The problems whose solution or algorithm for the solution can be determined in polynomial time complexity.

NP: Given a problem and a certificate, the problems whose certificates can be validated to be as solutions or not in polynomial time complexity fall under NP.

A problem Q is said to be in NP – Complete if

1. Q is in NP
2. For some other P in NPC, P is reducible to Q in polynomial time.

The first condition can be checked by verifying a given or guessed certificate of a problem to be its solution or not in polynomial time complexity.

The second condition is verified by taking an NPC problem and reducing it to the required problem in polynomial time complexity.

Sub - Graph Isomorphism problem: The graph isomorphism problem is one of the NP – Hard problems which neither belong to the class P nor NPC but is in NP but a problem derived from the graph isomorphism problem is an NP - Complete problem.

Graph isomorphism problem: Is a graph G1 isomorphic to graph G2??

Sub – graph isomorphism problem: Is a graph G1 isomorphic to subgraph of a graph G2??

Graph isomorphism problem is NP – Hard but Sub graph Isomorphism is NP – Complete.

Practical Applications:

1. In bioinformatics, it is always required to search an isomorphic special piece of protein sequence (3D labeled graph) in the Protein Data Bank (PDB) with the query protein sequence for further biological analysis.
2. In robot visions, it needs to recognize 3D objects (based on the line-drawings) by isomorphically matching with their canonical models in a computer model base.
3. In VLSI computer-aided design (CAD), one of the critical problems is that of determining whether layout of the circuit geometry corresponds to the specification of circuit, especially to find the sub-circuits from a larger circuit.

SGI problem is not only of considerable practical importance but also of theoretical interest due to its known NP-complete complexity. Its difficulty can be seen easily from the fact that selecting out of the mn possibilities that arise in the combinatorial matching of n nodes in the smaller graph to m nodes in the larger graph, while preserving all the adjacencies. No efficient algorithm for this problem is known so far, and it was conjectured by many experts that no polynomial-time algorithm exists because of its NP-completeness.

**Input:**

* Two graphs G and H are taken as input.

**Problem:**

* Is there a subgraph of G that is isomorphic to H???
* Let G = (V,E) and H = (V’,E’) be the graphs. Is there a sub-graph



* Therefore whentwo graphs *G*=(*VG*, *EG*) and *H=*(*VH*, *EH*) such that |*VH*|≦|*VG*| and |*EH*|≦|*EG*| are given, then thee task is to find out whether there is an injective mapping *f* from *VH* to *VG* such that {*f*(*u*), *f*(*v*)}∈*EG* holds for any {*u*, *v*}∈*EH*
* To see that the problem is in NP, we observe that a certificate is a mapping from the nodes of G1 to (a subset of) the nodes of G2, describing which vertices of G2 correspond to vertices of G1. The certifier then needs to make sure that for each edge e = (u, v) in G1, the edge (f(u), f(v)) is also in G2, and whenever (u, v) is not an edge of G1, then (f(u), f(v)) is not an edge of G2. This can be done with two simple nested loops, and takes at most O(n2) time, i.e., polynomial.
* We now show that Subgraph-isomorphism is NP-complete. We show this by using Subgraph-isomorphism to be reduced from the known NP-complete problem Clique.
* Let (G, k) be an instance of clique where we want to determine whether the graph G contains a clique of size k.
* We create the instance of Subgraph-isomorphism (G, Gk), where Gk is the complete graph with k nodes.
* We show that these two instances are equal. If G has a k-clique, then G has the complete k-graph as subgraph, and the instance (G, Gk) is also a yes-instance. If G has Gk as subgraph, then it has a k-clique as subgraph, and hence (G, k) is a yes-instance of Clique.
* We conclude that Subgraph-isomorphism is NP-hard and hence NP - complete.

**Knapsack Problem:**

Input:

* The instance of the well famous knapsack problem is a set of weights {w1, w2, …, wn }, b and a set of profits {c1, c2, …., cn}, k.

Problem:

* Is there a subset of the set of weights whose elements sum up to b and a subset of the cost/ profit set whose elements sum up to at least k.

Practical Applications:

1. Assume that n projects are available to an investor and that the profit obtained from the project is pj, j = 1…..n. It costs wj to invest in a project j and only c dollars are available. An optimal investment plan can be made using the knapsack problem.
2. If a person in a restaurant needs to choose k courses without surpassing the amount of c calories, his diet prescribes. Assuming that there are n dishes to choose for each course k and wij is the nutritive value while pij is a rating saying how well each dish tastes, then an optimal meal may be found out using the knapsack problem.
3. Apart from these simple illustrations, knapsack problem is used also in cargo loading, cutting stock, budget control, financial management etc.

A problem is in NP if a certificate can be checked whether it is the solution or not in polynomial time. The certificate in this problem will be two sets, one subset of the weight set and the other subset of the profit/cost set. Unless we have them, it is a very easy task to perform an addition operation to find out whether the sum of the elements of the first set sum up to b and the other is at least k in polynomial time, to be specific linear. Hence, we could undoubtedly conclude that the knapsack is in NP class complexity.

* In order to prove the np – completeness of the problem, we need to reduce it from another np – complete problem whose completeness has been proved, in polynomial time.
* Hence let us reduce it from Subset – sum problem because the instance of the problem is a set of non- negative elements (say si, i = 1,2 …, n) and a non – negative integer (say t) where the problem puts to find a subset of the given set whose elements sum up to the given number.
* Since, both of the problems relate to themselves having to find out a common property of subsets, we can use the Subset – sum problem to reduce it to an instance of the knapsack problem in polynomial time.
* If wi be the set of weights and ci be the set of costs, b be the given number for weights and k be the given number for costs. As a part of reduction, let us assume an instance of wi = ci = si and b = k = t.

Σ si = t

Σ ai ≤ b Σ si ≤ t

Σ ci ≥ k Σ si ≥ t

* Thus subset – sum problem has been reduced to an instance of the problem where the above transformation can be done in polynomial time.
* Suppose we have a Yes answer to the new problem, it means we can find such a subset S ⊆ [1, 2, · · · , n] where i ϵ S that satisfies the left part of the deduction. Then this subset S is also a solution to the right part. So we must also have a Yes answer to the original problem.
* Conversely, suppose we have a No answer, it means there is no subset S that satisfies the left part. So, of course, the answer to the original problem must also be No.
* Thus, the Knapsack problem is NP – Complete.

**Hamiltonian Cycle:**

* Hamiltonian cycle is such a set of vertices arranged in order such that the order represents a particular type of traversal of graph in which we start at a given vertex and attend every vertex exactly once and return to the same vertex again.
* For any arbitrary graph containing any number of vertices i.e. irrespective of the edges and vertices in the graph , the problem is to determine the possible Hamiltonian cycle from the graph.
* The decision version of the problem can be

Hamiltonian cycle

Given a directed graph G, does there exist a cycle that visits every vertex exactly once??

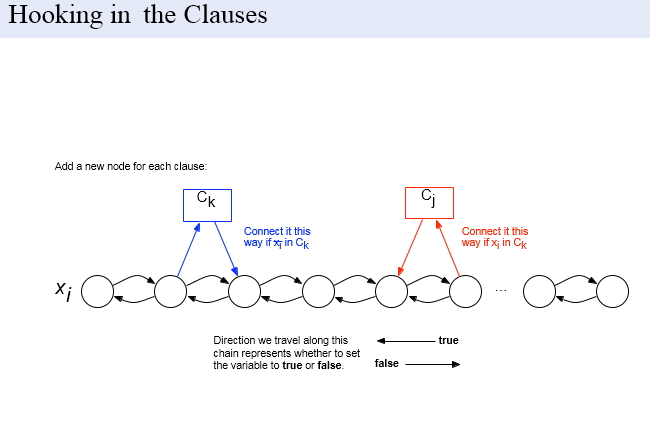
Let us reduce 3 – SAT problem to Hamiltonian Cycle.

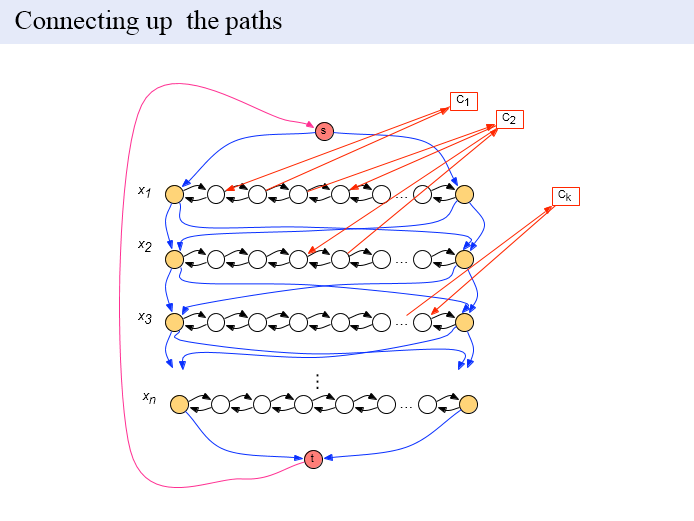
In order to reduce 3 – sat to Hamiltonian cycle, we need to create an instance of 3 – sat problem such that the instance becomes satisfiable only when the graphical representation of the problem contains a Hamiltonian cycle.

Reduction Idea:

Consider an instance of the 3 – sat problem consisting of n variables x1, x2, x3 …….. xn and k clauses c1, c2, c3 ……. ck.

Create a graph structure that represents the variables and clauses and let us show that the problem is satisfiable only when the graph has a Hamiltonian cycle.





A Hamiltonian cycle encodes a truth assignment for the variables (depending on which direction the chain is traversed).

For there to be a Hamiltonian cycle, there must be a truth assignment for each clause variable. But we can only visit a node only when there is a truth assignment to it. Therefore, if there is a Hamiltonian cycle for which we can visit all the nodes, then the problem becomes satisfiable. Hence, 3-sat is reduced to Hamiltonian cycle.

**Travelling Salesman Problem (TSP):**

* The travelling salesman problem can be described as follows:
* TSP = {

(G, f, t):

G = (V, E) a complete graph,

f is a function V×V → Z,

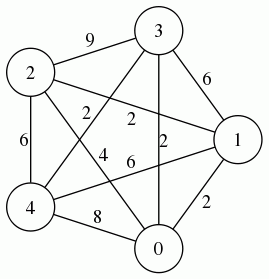
t ∈ Z, G is a graph that contains a traveling salesman tour with

cost that does not exceed t.

}

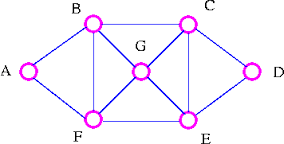
Practical Application:

* The travelling salesman problem consists of a salesman and a set of cities. The salesman has to visit each one of the cities starting from a certain one (e.g. the hometown) and returning to the same city. The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip.



For example, there are different paths in the graph starting from 1, reaching all vertices once and finally coming back to 1 such as 1 -> 3 -> 2 -> 4 -> 0 -> 1, 1 -> 2 -> 0 -> 3 -> 4 -> 1, 1 -> 3 -> 0 -> 4 -> 2 ->1 etc. but there exists a single path with the minimum cost.

* Each and every edge has some weight and the challenge is to find a path or to be specific, a circuit / cycle which has the shortest weight such that each vertex is visited only once.
* So the given certificate is a path and the minimum cost. We need to verify that the starting point and the ending point are same, all other vertices are visited only once, all the vertices are covered and the sum of weights of this path is equal to the minimum cost. This can be done in polynomial time complexity and hence the travelling salesman problem belongs to the NP class of complexity.
* Since we need to cover all the vertices only once and the starting point must be the same as the ending point, this reminds me of the Hamiltonian cycle. We have the Hamiltonian cycle problem as NP – Complete.
* Hence let us reduce the Hamiltonian cycle problem to Travelling Salesman Problem in polynomial time to prove the np – completeness of the problem.



The problem of finding a Hamiltonian cycle in a graph is NP – Complete. There are various Hamiltonian cycles in the given graph. For example A -> B -> C -> D -> E -> G -> F -> A is a Hamiltonian cycle or A -> B -> G -> C -> D -> E -> F -> A is another Hamiltonian cycle. Let us reduce this type of Hamiltonian cycle problem to travelling salesman problem in polynomial time.

* Assume G = (V, E) to be an instance of Hamiltonian cycle. An instance of TSP is then constructed. We create the complete graph G′ = (V, E′), where E′ = { (i, j) : for all i, j ∈ V and i ≠ j }. thus, the cost function is defined as:

1 , if ( i, j ) ∉ E

0 , if ( i, j ) ∈ E

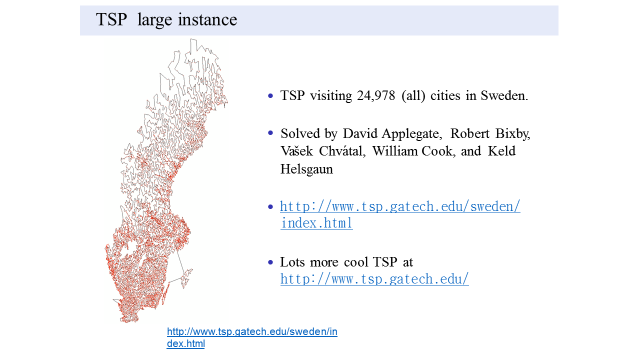
t =

Now suppose that a Hamiltonian cycle h exists in G. It is clear that the cost of each edge in h is 0 in Gas each edge belongs to E. Therefore, h has a cost of 0 in G′′. Thus, if graph G has a Hamiltonian cycle then graph G has a tour (path of tsp) of 0 cost.

Conversely, let us assume that G’ has a tour h’ of cost at most 0. The cost of edges in E’ are 0 and 1 by definition. So each edge must have a cost of 0 as the cost of h’ is 0. We conclude that h’ contains only edges in E. So we have proven that G has a Hamiltonian cycle if and only if G’ has a tour of cost at most 0.

Hence, to have an at most zero cost of a path for travelling salesman problem, we need to have an Hamiltonian cycle thereby reducing Hamiltonian cycle to an instance of travelling salesman problem in polynomial time.

Thus the travelling salesman problem is NP - Complete.



**Quadratic Assignment Problem:**

Problem:

For any two given sets, P and Q of equal size and two functions – weight function and distance function such that w: P x P R and d: L x L R. We need to find a bijection f: P L such that the cost function:

Σw(a, b) . d(f(a), f(b))

a,b ϵ P

is minimized.

Practical Application:

There are a set of n facilities or factories and a set of n locations. For each pair of locations, the distance between the locations is given and for each pair of facilities, the weight or flow is specified (the amount of trade between the facilities) is given. The problem is to assign all facilities to different locations ( a unique facility to a unique location) with the goal of minimizing the sum of the product of distances of locations and flow between the facilities.

Intuitively, the facilities which have large amount of flow must be placed closed to each other.

* We may think that even if the minimized value is given, to check whether it is minimum or not of the all possible combinations takes exponential time. Here’s where we are wrong.
* We will have both the function and the minimum value. We need to check the cost function value obtained through the function is greater than or less than the given minimum value i.e. Σw(a, b) . d(f(a), f(b)) ≤ k where k and f are given has to be checked which obviously takes a polynomial time.
* Hence, the problem is in NP.
* To prove that the problem is np-complete, let us show that the TSP reduces to QAP. Here is a decision version of TSP :
* For a given set V with a distance metric d(a, b), a, b ∈ V , and a number k, is there such an ordering of elements of V : (v1, v2, ..., vn, v1) that the following is true:

Σ

d(vi , vi+1) + d(vn, v1) ≤ k

1≤i<n

* To solve this TSP, let’s formulate the QAP. We’ll have a set P of n facilities ordered in a linear sequence: P = (p1, p2 ...pn). The flow between two sequential facilities is 1 and between any other facilities is 0 i.e. w(pi , pj ) = 1, if (i - j mod n) = 1; 0 otherwise. And a set L is equal to the set V with the same distance metric.
* Thus we have interpreted the travelling salesman problem as an instance of Quadratic Assignment Problem thereby concluding the problem to be NP – Complete.