Module 12
Introduction to Complexity Theory

S. Lakshmivarahan
School of Computer Science
University of Oklahoma
USA-73019
Varahan@ou.edu

Tractable Vs Intractable Problems

- All the problems solved in this course so far take time of the form log n, n, n log n, n^2, n^3, etc.
- Each of this time functions are polynomially bounded.
- Definition: The set of all problems solvable in polynomially bounded time, are called Tractable Problems
- Problems that require exponential time, 2^n are known as Intractable.

Some Intuition

- If a “LARGE” sized problem can be solved in a “Short” time, it is Tractable.
- Intractable problems are those require “LARGE” time to solve even “SMALL” sized problems.
- Problem π_1 of size n_1 with $T_1(n)=n_1^3$

  Problem π_2 of size n_2 with $T_2(n)=2^{n_2}$

Intuition- continued

- Let $n_1=10^6$ then $T_1(n_1)= 10^{18}$ operations
- Verify: $n_1^3 = 10^{18} = 2^{n_2}$ and $n_2 = 59.78$
- Problem π_2 of size merely $n_2 = 60$, takes the same time as problem π_1 of size $n_1=10^6$
- π_1 is tractable, π_2 is not.

Prevalence of Intractable Problems

- A large variety of combinational problems from many walks of life are intractable
- Some of these are “DECISION PROBLEMS” who’s answer is YES or NO
- A large variety are “Optimization Problems”
- Complexity theory is built around the decision problems.
- In assessing the complexity of optimization problems, consider the decision version
- The idea is if the decision version is difficult, the optimization is even more difficult.

Example - Satisfiability Problem

- X- Logical Variable – Takes two values : T or F
- $\bar{X}$ - Complement of X
- $\{X, \bar{X}\}$ – Literal
- A clause is a disjunction (Boolean "OR", ∨) of literals
- $(X_1 \vee X_2 \vee X_3)$, $(\bar{X}_1 \vee X_2 \vee \bar{X}_3)$ $(\bar{X}_1 \wedge X_3)$ are clauses
- A Logical Expression is in conjunctive normal form if it a conjunction (Boolean "AND", ∧) of clauses
- $f(x_1, x_2, x_3) = (\bar{X}_1 \vee X_3) \wedge (\bar{X}_1 \vee X_2 \vee \bar{X}_3) \wedge (X_1 \vee X_2 \vee X_3)$ is a Logical Expression

Satisfiability (SAT) Problem - decision problem

- Set x_1 = F, x_2 = F, x_3 = T – called truth assignment for logical variables x_1, x_2, x_3.
- Verify that $f(x_1, x_2, x_3)$ is "TRUE" for this truth assignment – Satisfying Assignment
- Given a logical expression $f(x_1, x_2, \ldots . x_n)$ finding an assignment of truth values to $x_1, x_2, \ldots . x_n$ that makes $f(x_1, x_2, \ldots . x_n)$ True is called the satisfiability problem.
- This is combinatorial decision problem since there are 2^n distinct truth assignments to $x_1 \ldots . x_n$.
- In the worst case, have to exhaust all the 2^n possibilities – Intractable

SAT Problem

- Naturally arises in automatic theorem proving.
- Played a central role in the development of “COMPLEXITY THEORY” as we know it today.
- An intimate relation between SAT problem and algorithms implemented on Turing machines.
- It is a model for Intractable Decision Problem.

Final Exam time table problem

- A large university wants to schedule all the final exams in a week with four times slots of two hours each on a given day with a total of twenty time slot.
- Constraints – Student taking different subjects must have exam in different time slots.
- Verify: This is a combinatorial decision problem – Does there exits a schedule satisfying the constraint?

Reduction to graph coloring problem

- Define G=(V, E)
- V – The set of all courses {1, 2 , …..n} in the university
- E – Two courses i and j are adjacent if they cannot have exam at the same time.
- The number of colors are equal to the total number (=20) of distinct time slots available.
- The final exams can be scheduled without conflicts only if we can color the nodes of the graph in such a way that no two adjacent nodes have the same color – called Legal coloring of the graph
- Decision Version: Is there a legal coloring of a graph with atmost k – colors (k = 20) in time table problem.
- Optimization Problem: Find the chromatic number $\chi(G)$ the smallest number of colors needed to legally color a graph.

Hamiltonian cycle (HC) and travelling salesman problem (TSP)

- Let G = (V, E) be a graph.
- HC is the simple, spanning cycle.
- Let G = (V, E, W) be a weighted graph with positive weights.
- Optimization- TSP: Find a tour (which is a HC) of least cost.
- Decision – TSP: Is there a tour of cost at most k ?
- Applications : routing trucks through a city to pick up garbage, deliver package

Tractability and P - class

- A problem π is said to be in class P (called the polynomial class) if there exits a polynomial time algorithm $T(n) = O(n^k)$, $k>0$ integer, to solve π
- Tractability is identified with P – class
- P is the class of polynomially bounded decision problems.

Properties of P - class

- Tractability or P – class => Efficient algorithm
- P – Class has nice closure properties.
- P- Class closed under addition, multiplication, and composition of complexities of component algorithms.
- No smaller class has this closure property.
- The P – class is independent of the formal model for computation – Turing machine models, RAM model, circuit model, etc.
- If a problem π has polynomial time in one model then it enjoys polynomial time in all other models.

Intractability and NP-class

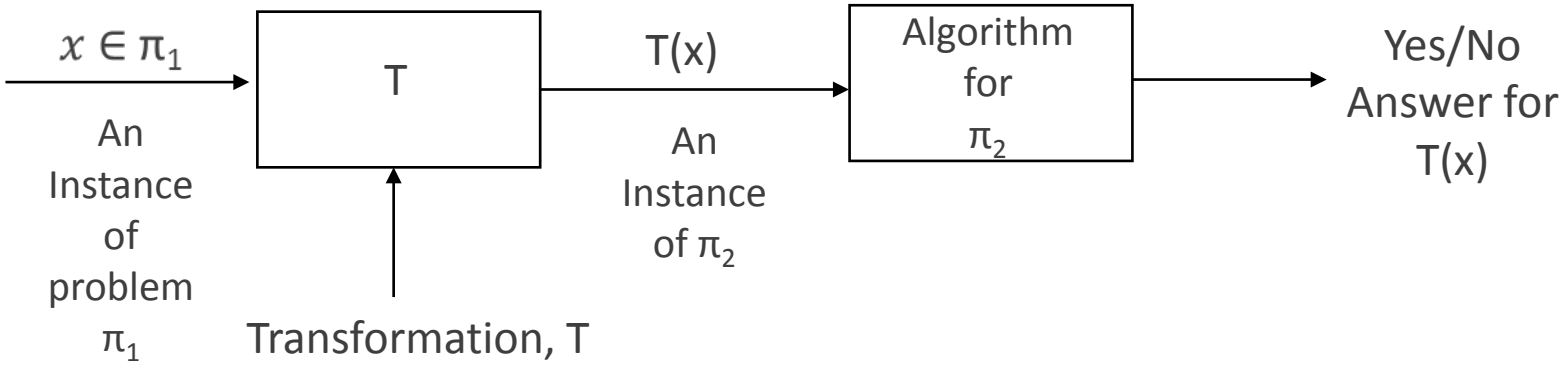
- NP – Non deterministic polynomial class
- A problem π in NP admits a non deterministic polynomial time algorithm on a Turing machine.
- A non- deterministic finite automata with n – states can be transformed into a deterministic finite automata with 2^n states.
- Likewise, a non – deterministic polynomial time algorithm takes exponential time when implemented deterministically.

Properties of NP-Class

- NP algorithm has two phases.
- Non- deterministic/ Guessing phase: Guess a solution
- Deterministic phase: Verify if the suspected solution is indeed a solution.
- Example: Pick an arbitrary truth assignment from among 2^n assignments.
- Verify if the given assignment is a satisfying assignment.

Complete problems- transformations

- Let L be a class of problems.
- A problem $\pi \in L$ is called a Complete Problem for the class L if every problem $\pi' \in L$ can be reduced or transformed to the problem π denoted by $\pi' \rightarrow \pi$
- We are interested in transformation or reduction of π_1 to π_2

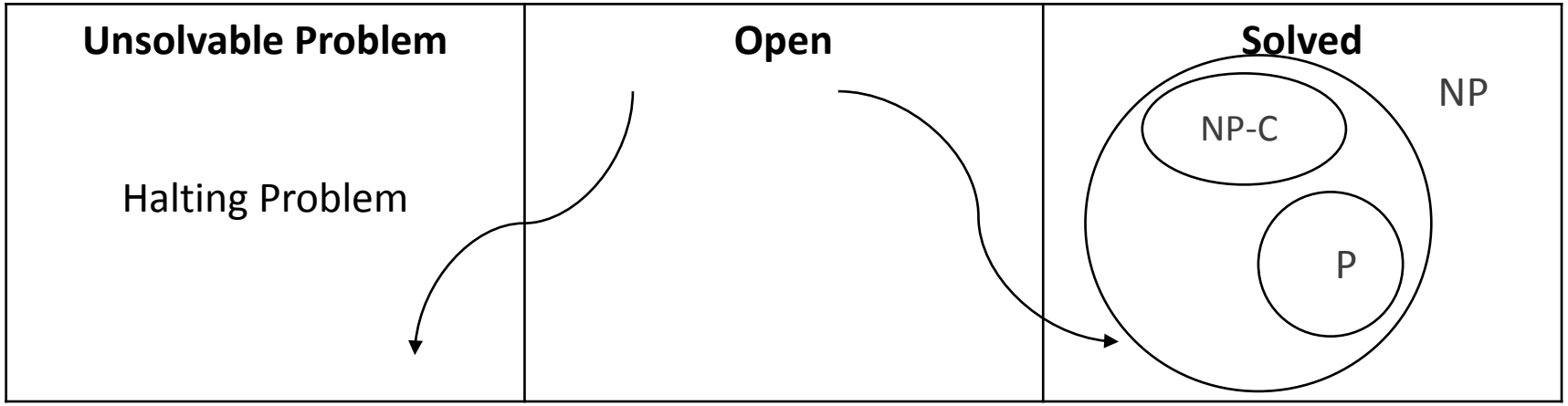


Polynomial transformation NP- Completeness

- Motivated by feasibility, interested in polynomial transformations
- Problem π is a complete problem for the NP Class if $\pi \in$ NP and for every problem $\pi' \in$ NP, there is a polynomial time transformation / reduction denoted by $\xrightarrow{P}$ such that $\pi' \xrightarrow{P} \pi$.
- S Cook in 1971 first proved that the SAT problem is NP- complete (ie) SAT is NP-C

Frame work – The complexity of combinational problems

- Let $\pi_1 \xrightarrow{P} \pi_2$ and $\pi_2 \in P$, then $\pi_1 \in P$.
- To prove that a problem π is NP– C, select a known NP-C problem π' and produce a polynomial time transformation such that $\pi' \xrightarrow{P} \pi$.
- A view of the world of problems



- Problem: Argue P ⊆ NP

IS P=NP ?

- It is not known if P=NP and is a major open problem.
- To show P=NP, enough to show that a known NP-C problem can be solved in polynomial time (ie) for all $\pi' \in NP$.

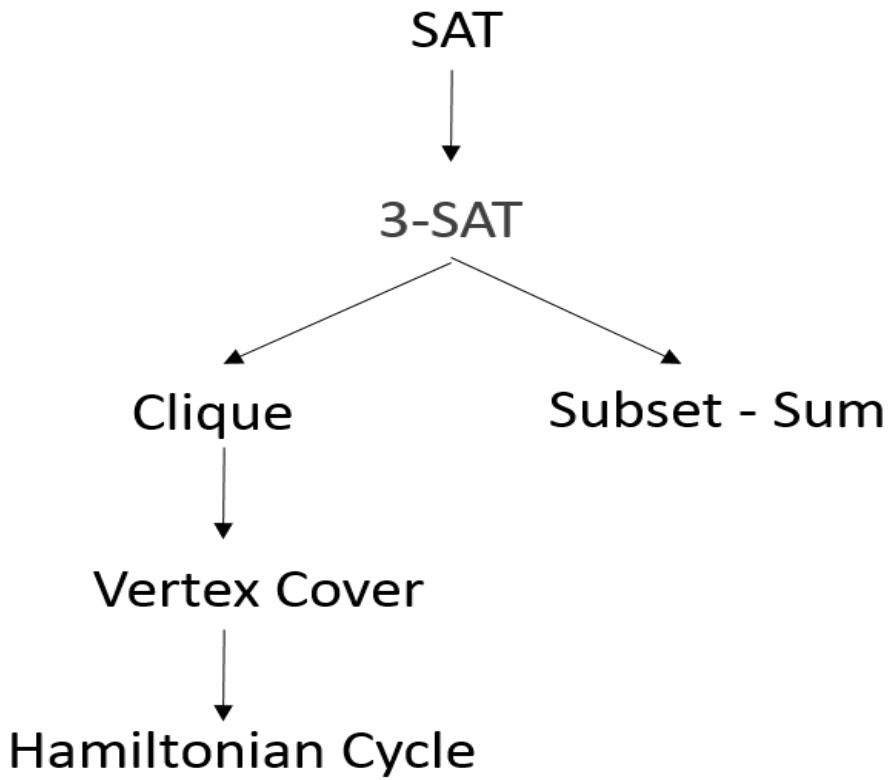
$$\pi' \xrightarrow{P} \pi \in P$$

- Hence every problem is NP can be solved in polynomial time and hence NP =P
- This is one of the major open problems of our time.

NP- hard Problems

- By definition, NP is the class of decision problems.
- A combinational optimization problem eg : TSP cannot be in NP
- But the decision version of it is in NP and can be shown to be NP-C.
- Thus, any combinatorial optimization problem whose decision version is NP-C is called NP-hard. TSP is NP-hard.

Tree of NP-C problems



- This tree is very big and grows continuously as new NP –C problems are discovered