Chapter 5

Theory of NP

The spectrum of computational complexity

- We have seen many problems which can be solved by **polynomial-time algorithms**: on inputs of size n, their worst-case running time is $O(n^k)$ for some constant k.
- We have some **undecidable (unsolvable)** problems: cannot be solved by any computer, no matter how much time we allow.
  - e.g., Turing halting problem
- There are also problems that can be solved, but not in time $O(n^k)$ for any constant k: **superpolynomial time, intractable, or hard problems**
  - e.g., List all permutations of n, Towers of Hanoi numbers
- The subject of this chapter, however, is an interesting class of problems, called the **NP-complete** problems, whose status is unknown.

Background: Reduction

The Reduction Joke:
Professor X, a noted mathematician, noticed that when his wife wanted to boil water for their tea, she took their kettle from their cupboard, filled it with water, and put it on the stove.
Once, when his wife was away, the professor had to boil water by himself. **He saw that the kettle was sitting on the kitchen counter.**
What did Professor X do? He put the kettle in the cupboard first and then proceeded to follow his wife's routine.☺

Consider decision problems, which are problems with yes/no answers.

Definition (Reduction)

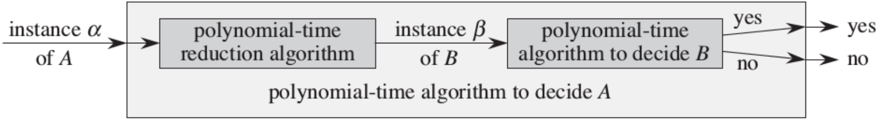
A decision problem *D1* is said to be polynomially reducible to a decision problem *D2*, if there exists a function t that transforms instances of *D1* to instances of *D2* such that:

- t maps all yes instances of *D1* to yes instances of *D2* and all no instances of *D1* to no instances of *D2*
- t is computable by a polynomial time algorithm

We can represent this relationship by this notation: $D1 \leq_p D2$

This definition immediately implies that

1. If *D2* that can be solved in polynomial time, then problem *D1* can also be solved in polynomial time.
2. If *D1* cannot be solved in polynomial time, then *D2* cannot be solved in polynomial time.



How to use a polynomial-time reduction algorithm to solve a decision problem A in polynomial time, given a polynomial-time decision algorithm for another problem B.

P, NP, NP-Complete

For the decision problems, we can define the following classes:

1. **Class Polynomial (P):** can be "solved" in polynomial time.
2. **Class Nondeterministic Polynomial (NP):** are "verifiable" in polynomial time: if we were somehow given a "certificate" of a solution, then we could verify that the certificate is correct in time polynomial in the size of the input to the problem.

Example

In the hamiltonian-cycle problem, given a directed graph $G = (V, E)$, a certificate would be a sequence $v_1, v_2, \cdots, v_{|V|}$ of $|V|$ vertices. We could easily check in polynomial time that $(v_i, v_{i+1}) \in E$ for $i = 1, 2, \cdots, |V| - 1$ and that $(v_{|V|}, v_1) \in E$ as well.

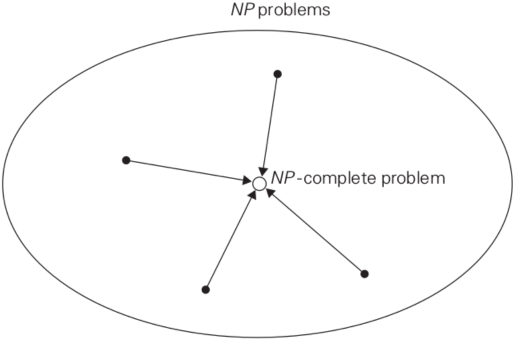
- we have (WHY?)

$$P \subseteq NP$$

3 **Class NP-Complete (NPC):** A decision problem D is said to be NP-complete if:

- it belongs to class NP
- **every** problem in NP is polynomially reducible to D

The notion of NP-completeness requires polynomial reducibility of **all** problems in NP, both known and unknown, to the problem in question.



- 1971, independently by **Stephen Cook** in the United States and **Leonid Levin** in the former Soviet Union–**CNF-satisfiability problem is NP-complete**
    - CNF-satisfiability problem: each boolean expression can be represented in conjunctive normal form (ANDs of ORs), such as $(x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee x_2) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$. The CNF-satisfiability problem asks whether or not one can assign values true and false to variables of a given boolean expression in its CNF form to make the entire expression true (for the above formula: if $x_1 = true$, $x_2 = true$, and $x_3 = false$, the entire expression is $true$.).

- In his paper, Cook also introduced the famous **P versus NP problem (is P=NP ?)**.
  - A large majority of theoretical computer scientists believe that P $\neq$ NP, which would mean that no NP-complete problem can be solved in polynomial time.
  - The P versus NP problem is one of the most famous unsolved problems in the mathematical sciences (which include theoretical computer science). It is one of the seven famous Millennium Prize Problems, of which six remain unsolved. A prize of $1,000,000 is offered by the Clay Mathematics Institute for its solution.

STEPHEN COOK (BORN 1939) Stephen Cook was born in Buffalo where his father worked as an industrial chemist and taught university courses. His mother taught English courses in a community college. While in high school Cook developed an interest in electronics through his work with a famous local inventor noted for inventing the first implantable cardiac pacemaker.

Cook was a mathematics major at the University of Michigan, graduating in 1961. He did graduate work at Harvard, receiving a master's degree in 1962 and a Ph.D. in 1966. Cook was appointed an assistant professor in the Mathematics Department at the University of California, Berkeley in 1966. He was not granted tenure there, possibly because the members of the Mathematics Department did not find his work on what is now considered to be one of the most important areas of theoretical computer science of sufficient interest. In 1970, he joined the University of Toronto as an assistant professor, holding a joint appointment in the Computer Science Department and the Mathematics Department. He has remained at the University of Toronto, where he was appointed a University Professor in 1985.

Cook is considered to be one of the founders of computational complexity theory. His 1971 paper "The Complexity of Theorem Proving Procedures" formalized the notions of NP-completeness and polynomial-time reduction, showed that NP-complete problems exist by showing that the satisfiability problem is such a problem, and introduced the notorious P versus NP problem.

Cook has received many awards, including the 1982 Turing Award. He is married and has two sons. Among his interests are playing the violin and racing sailboats.

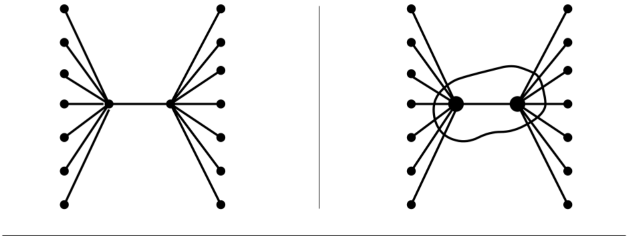
Every NP is reducible to SAT

Example

Reduction of Vertex Cover to SAT

VERTEX-COVER

- Input: An undirected graph $G = (V, E)$, and an integer $k \in [0, |V|]$.
- Output: True, if G has a (vertex-cover) set of k vertexes which cover all edges. An edge is covered if either of its two end nodes belongs to the cover set.



CNF-SAT is the following problem:

- Input: a propositional logic formula, F, in Conjunctive Normal Form (CNF).
- Output:True, if F is satisfiable, false otherwise.

The reduction

Given $< G, k >$ where $G = (V, E)$, denote $|V| = n$. Assume that the vertices are named $1, ..., n$. Construct F as follows.

1. Variables:
   Adopt $n \times k$ atoms, each denoted $x_{i,j}$ where $i \in [1, n]$ and $j \in [1, k]$. The mindset behind this is the following. We see a vertex cover of size k as a list of k vertices. The meaning of $x_{i,j}$ is that $x_{i,j}$ is true if and only if the vertex i is the jth vertex in the vertex cover.

2 Clauses

- $\forall j \in \{1, \cdots, k\}$, a clause $x_{1,j} \vee x_{2,j} \vee \ldots \vee x_{n,j}$. why?
- $\forall i \in \{1, \cdots, n\}$, $\forall j_1, j_2 \in \{1, \cdots, k\}$ with $j_1 < j_2$, a clause $\bar{x}_{i,j_1} \vee \bar{x}_{i,j_2}$. why?
- $\forall j \in \{1, \cdots, k\}$, $\forall i_1, i_2 \in \{1, \cdots, n\}$ with $i_1 < i_2$, a clause $\bar{x}_{i_1,j} \vee \bar{x}_{i_2,j}$. why?
- For each edge $(i, j) \in E$, a clause $x_{i,1} \vee x_{i,2} \vee \ldots \vee x_{i,k} \vee x_{j,1} \vee x_{j,2} \vee \ldots \vee x_{j,k}$. why?

► The number of clauses that the reduction outputs is therefore:

$k + n \binom{k}{2} + k \binom{n}{2} + |E|$.

Remaining work: We must show that this transformation is a reduction.... **left as an exercise**

Therefore, having a SAT solver is a generic tool to solve all NP problems.

- Competitions to find efficient SAT solvers

www.satcompetition.org

The international SAT Competitions web page

Current Competition

SAT 2019 Race	
Organizers	Marijn Heule, Matti Järvisalo, Martin Suda

Past Competitions

SAT 2018 Competition	
Organizers	Marijn Heule, Matti Järvisalo, Martin Suda

SAT 2017 Competition	
Organizers	Marijn Heule, Matti Järvisalo, Tomáš Balyo
Slides	Slides used at SAT 2017
Proceedings	Descriptions of the solvers and benchmarks
Benchmarks	Available here
Solvers	Available here

	Gold	Silver	Bronze	Gold	Silver	Bronze	Gold	Silver	Bronze
	Agile Track			Main Track			Random Track		
SAT+UNSAT	CaDiCaL Agile, CaDiCaL NoProof	Glu_VC	Glucose 4.1	Maple LCM Dist, Maple LCM, MapleLRB LCMOccRestart, MapleLRB LCM	MapleCOMSPS LRB VSIDS 2, MapleCOMSPS LRB VSIDS	COMiniSATPS Pulsar	YalSAT	tch glucose3	Score2SAT
	Parallel Track			No-Limit Track			Incremental Library Track		
SAT+UNSAT	Syrup24, Syrup48	Plingeling	Painless MapleCOMSPS	COMiniSATPS Pulsar	MapleCOMPSPS LRB VSIDS 2, MapleCOMPSPS LRB VSIDS	CaDiCaL NoProof	AbcdSAT	Glucose	Riss

SAT 2016 Competition	
Organizers	Marijn Heule, Matti Järvisalo Tomáš Balyo
Proceedings	Descriptions of the solvers and benchmarks

- Since the Cook-Levin discovery of the first known NP-complete problems, computer scientists have found thousands of other examples. Because of the following observation:

If Y is an NP-complete problem, and X is a problem in NP with the property that Y $\leq_P$ X, then X is NP-complete.

  - For instance, Hamiltonian circuit, traveling salesman, partition, bin packing, and graph coloring are all NP-complete.
- Proving that a problem is NPC, can
  - save your time not trying to find polynomial algorithm
  - save your job when discussing with your boss
  - be an important theoretical achievement in your research publication

- **Hamiltonian circuit problem** Determine whether a given graph has a Hamiltonian circuit–a path that starts and ends at the same vertex and passes through all the other vertices exactly once.
- **Traveling salesman problem** Find the shortest tour through n cities with known positive integer distances between them (find the shortest Hamiltonian circuit in a complete graph with positive integer weights).
- **Partition problem** Given n positive integers, determine whether it is possible to partition them into two disjoint subsets with the same sum.
- **Bin-packing problem** Given n items whose sizes are positive rational numbers not larger than 1, put them into the smallest number of bins of size 1.
- **Graph-coloring problem** For a given graph, find its chromatic number, which is the smallest number of colors that need to be assigned to the graph's vertices so that no two adjacent vertices are assigned the same color.

Example: Reduction of Hamiltonian cycle to TSP

This is the Hamiltonian cycle problem:

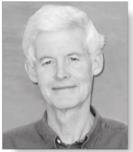
Problem: Hamiltonian Cycle
Input: An unweighted graph G.
Output: Does there exist a simple tour that visits each vertex of G without repetition?

The reduction from Hamiltonian cycle to traveling salesman:

HamiltonianCycle$(G = (V, E))$
    Construct a complete weighted graph $G' = (V', E')$ where $V' = V$.
    $n = |V|$
    for $i = 1$ to n do
        for $j = 1$ to n do
            if $(i, j) \in E$ then $w(i, j) = 1$ else $w(i, j) = 2$
    Return the answer to Traveling-Salesman-Decision-Problem(G', n).

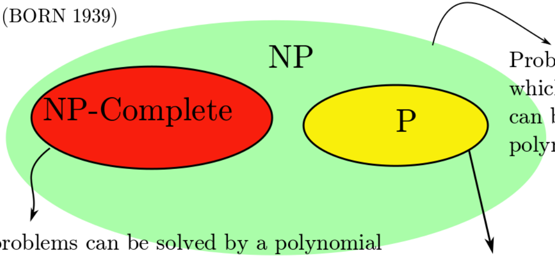
Proof of the reduction correctness:
If the graph G has a Hamiltonian cycle, then this exact same tour will correspond to n edges in E', each with weight 1. This gives a TSP tour in G' of weight exactly n. If G does not have a Hamiltonian cycle, then there can be no such TSP tour in G' because the only way to get a tour of cost n in G would be to use only edges of weight 1, which implies a Hamiltonian cycle in G.

Summery



The P versus NP problem asks whether NP equals P. If P is not equal to NP, there would be some problems that cannot be solved in polynomial time, but whose solutions could be verified in polynomial time.

STEPHEN COOK (BORN 1939)



Problems for which a solution can be checkedin polynomial time

if any of these problems can be solved by a polynomial worst-case time algorithm,then all problems in the class NP can be solved by polynomial worst-case time algorithms.

More than 3000 NP-complete problems are now known: SAT, Knapsack, TSP, Hamilton Circuit, Vertex Cover, Set Cover, ...

solvable using an algorithm with polynomial worst-casecomplexity (also called tractable)