

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**.

برای بعضی از مسئله ها پیچیدگی چند جمله ای پیدا شده و به مسائلی میگویم polynomial-time algorithms که وقتی تحلیل میکنیم پیچیدگیشونو از بیگ او یک چند جمله ای است

اما بعضی از مسئله ها حل شدنی نیستند مثل مسئله تورینگ

یه دسته از مسائل هستند که به هیچ وجه همیشه برای اونا پیچیدگی چند جمله ای ارائه کرد مثل

List all permutations of n و برج هانوی

یک دسته از مسائل هستند که وضعیتشون مشخص نیست یه np-complete یه نمیدونیم اینا جز مسائلی هستند که براشون الگوریتم چندجمله ای وجود داره یا نداره

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. 😊**

-
اگر مسئله جدید رو به مسئله ای تبدیل کنیم که جوابشو از قبل میدونیم بعد میگیریم جواب اون مسئله رو بلدیم و اینطوری مسئله جدید حل میشه این میشه reduction
ینی یک مسئله ای رو حلشو بلدیم و در مواجهه با مسائل جدید میایم بعد یه کاری میکنیم که اون به مسائلی که حلشو بلدیم تبدیل بشه و بعد طبق اون چیزی که بلدیم جلو می ریم

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.

مشکلات تصمیم‌گیری را در نظر بگیرید، که مشکلاتی با پاسخ‌های بله/خیر هستند. : خیلی مسائل رو می‌شه نسخه بله و خیر براشون ساخت پس ما مسائل رو نسخه decision در نظر می‌گیریم ینی خروجی‌شون یا بله است یا خیر
تعریف (کاهش)

اگر یک تابع t وجود داشته باشد که نمونه‌های $D1$ را به نمونه‌های $D2$ تبدیل میکند، به صورت چندجمله‌ای به یک مسئله تصمیم‌گیری $D2$ تقلیل پذیر گفته میشود: $-->$ فرض میکنیم یک مسئله ای مثل $D1$ داریم و $D1$ به صورت چند جمله ای قابل کاهش است به $D2$ اگر شرط‌های زیر باشد:

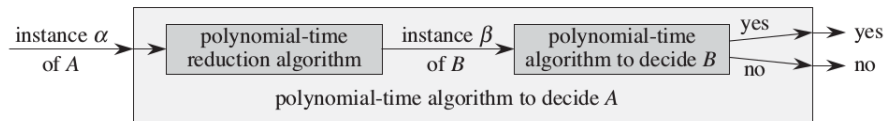
t همه موارد بله از $D1$ را به موارد بله از $D2$ و همه موارد خیر از $D1$ به موارد خیر از $D2$ نگاشت می‌کند. $-->$ تمام نمون‌هایی که جوابشون بله میشه توی مسئله $D1$ تبدیل بشن به نمونه‌هایی که جوابشون بله است توی نمونه $D2$ ینی اگر جواب بله است بله بمونه و اگر خیر است همین خیر بمونه

t با یک الگوریتم زمان چند جمله ای قابل محاسبه است $-->$ t به صورت چند جمله ای این کارو انجام بده ینی تبدیل $D1$ به $D2$ به صورت چند جمله ای بتونه انجام بشه در این صورت می‌گیم $D1$ قابل کاهش به $D2$ است و با نماد $D1 \leq_p D2$ نشون میدن

ما می‌تونیم این رابطه را با این نماد نشان دهیم: $D1 \leq_p D2$

این تعریف بلافاصله به آن اشاره دارد--> پس از موارد بالا می‌تونیم نتیجه بگیریم که:

- 1- اگر $D2$ که در زمان چند جمله ای قابل حل است، مسئله $D1$ نیز در زمان چند جمله ای قابل حل است. : اگر $D2$ به صورت چند جمله ای قابل حل باشه می‌تونیم نتیجه بگیریم مسئله $D1$ به صورت چند جمله ای قابل حل است
 - 2- اگر $D1$ را نمی‌توان در زمان چند جمله ای حل کرد، $D2$ را نمی‌توان در زمان چند جمله ای حل کرد. : نکته دوم مهم‌تره: اگر $D1$ قابل حل به صورت چند جمله ای نباشد $D2$ هم قابل حل به صورت چند جمله ای نخواهد بود اثباتش با برهان خلف است :
- فرض میکنیم $D2$ به صورت چند جمله ای قابل حل است و طبق نکته اول اینجا $D1$ هم میشه چند جمله ای و اینجا به تناقض می‌رسیم



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 .

نحوه استفاده از الگوریتم کاهش چند جمله ای زمان برای حل مسئله تصمیم A در زمان چند جمله ای، با توجه به الگوریتم تصمیم گیری چند جمله ای برای مسئله دیگر B

اگر نمونه های مسئله A توسط یک الگوریتم چند جمله ای کاهش بدیم به نمونه های مسئله B ینی تبدیلهشون کنیم به B که برای B الگوریتم چند جمله ای داریم و از این می تونه استفاده برای اینکه جواب به نمونه های مسئله A به دست بیاد مثلا اگر خروجی بله باشه جواب الفا بله است و اگر خیر است جواب الفا خیر است پس نمونه های مسئله A به صورت چند جمله ای قابل حل است

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 $v_1, v_2, \dots, v_{|V|}$ of $|V|$ vertices. We could easily check in polynomial time that $(v_i, v_{i+1}) \in E$ for $i = 1, 2, \dots, |V| - 1$ and that $(v_{|V|}, v_1) \in E$ as well.

► we have (WHY?)

$$P \subseteq NP$$

برای مسائل تصمیم گیری، می توانیم کلاس های زیر را تعریف کنیم:

1- کلاس چند جمله ای (P): می تواند در زمان چند جمله ای "حل" شود.

2- کلاس چند جمله ای غیر قطعی (NP): در زمان چند جمله ای "قابل تایید" هستند: اگر به نحوی یک "گواهی" از یک راه حل به ما داده شود، می توانیم تأیید کنیم که گواهی در چند جمله ای زمانی در اندازه ورودی صحیح است. مسئله. --> به مسائلی که به صورت چند جمله ای می تونه جوابشون تایید بشه ینی اگر یک نمونه از این مسائل داشته باشیم که جوابش بله میشه و بعد یک یک سولوشون به ما بدن و بگن این مسئله جوابش بله است و بگن سولوشون هم بله است و اگر ما بتونیم سولوشون رو چک کنیم تایید کردن این به صورت چند جمله ای قابل انجام شد توی مسائل NP

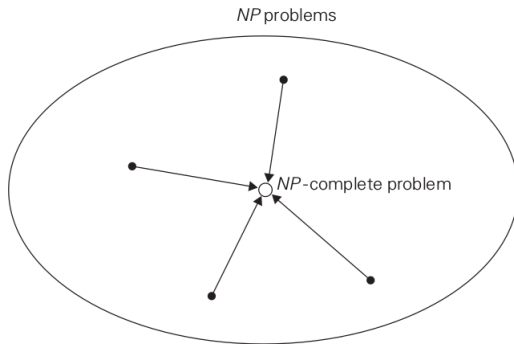
توی این مسئله می خوام ببینم یک سیکلی وجود داره که از همه رئوس عبور کنه این یک مسئله

NP-complete است ینی تا حالا الگوریتم چند جمله ای براش ارائه نشده اما یک مسئله NP است چون اگر یک گرافی داشتیم که جوابش بله بود و سولوشون هم بهم داده واسه این که اینو اثبات کنیم اول باید ببینم بین اون راس هایی متوالی که گفته یال وجود داره یا نه واز طرفی ببینم اون راس هایی که گفته تمام رئوس ما هستند یا نه و بیایم اینو دیگه بررسی کنیم و به این هم توجه کنیم که ز راس اخر یک یالی به راس اول وجود داشته باشه --> این کارو به صورت چند جمله ای می تونیم انجام بدیم

نکته: P ما زیرمجموعه NP است چون اگر یک نمونه بله از مسئله چند جمله ای که قابل حل است به ما بدن و ما باید به صورت چند جمله ای تایید کنیم اینو و نکته ای که هست P ها میتونیم به certificate نگاه نکنن و خود مسئله رو به صورت چند جمله ای حل بکنیم

- 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.



اولاً NPC جز مجموعه NP است و همه مسائلی که NP هستند قابل کاهش به NPC هستند به صورت چند جمله ای
پس اگر NPC رو به صورت چند جمله ای بتونیم حل بکنیم میتونیم ادعا کنیم همه مسائل NP به صورت چند جمله ای قابل حل هستند

- 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 $x_1 = \text{true}$, $x_2 = \text{true}$, and $x_3 = \text{false}$, the entire expression is *true*.).

CNF اولین مسئله ای است که جز NPC بود
سوالی که هست اینه که آیا برای این متغیرها مقداری وجود داره که جواب نهایی ما بله بشه یا نه
پس همه مسائلی که NP هستند رو می شه تبدیل کرد به مسئله CNF ینی کاهش داد

- 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.

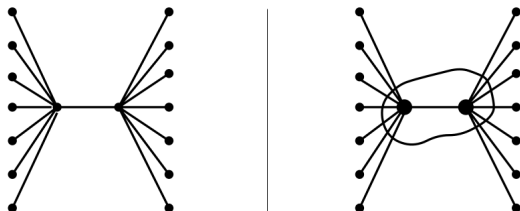
Every NP is reducible to SAT

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.



همه مسائل NP قابل کاهش به CNF هستند یک مثال برایش می زنیم پایین:

کاهش vertex cover به مسئله SAT

یک vertex cover به اندازه دو داریم که اگر اون دوتا راسو بگیریم همه یال هارو پوشش میده

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 $\langle G, k \rangle$ where $G = (V, E)$, denote $|V| = n$. Assume that the vertices are named $1, \dots, n$. Construct F as follows.

1 Variables:

اینو میخوایم تبدیل کنیم به F یکنی نمونه این مسئله تبدیل بشه به نمونه اون مسئله
برای این کار یه سری متغیر می گیریم که پایین گفته:

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 j th vertex in the vertex cover.

حالا میخوایم مسئله رو کاهش بدیم

مسئله **vertex cover** ما یک NP است چون اگر به ما بگن ایا **vertex cover** با سایز دو وجود داره و به من بگن بله و به ما **certificate** میدن و چک کردن این **certificate** به صورت چند جمله ای انجام میشه کافیه ما روی مجموعه یال ها حرکت کنیم و ببینیم یکی از این دو راس یال داخل این مجموعه که گفته ینی **certificate** داره هست یا نیست که این کار هم به صورت چند جمله ای انجام میشه

می خوایم اون الگوریتم چند جمله ای رو پیدا کنیم چجوری؟ یک نمونه از مسئله **vertex cover** رو تبدیل کنیم به این نمونه از مسئله CNF به نحوی که اگر این طرف نمونه شد بله اون طرف هم به نمونه بله CNF برسیم و اگر خیر بود اونطرف هم خیر باشه

شیوه کار: زرده میشه

G میشه گراف ما و K میشه **vertex cover**

تعریف متغیر: یکسری متغیر اول کار می سازیم و متغیرهایی که باید در نظر بگیریم $n \cdot k$ است و اسمشون رو $x_{i,j}$ می داریم

$x_{i,j}$ یک است اگر و تنها اگر i vertex ام j امین vertex در **cover** باشه و در غیر اینصورت صفر

2 Clauses

- $\forall i \in [1, k]$, a clause $x_{1,i} \vee x_{2,i} \vee \dots \vee x_{n,i}$. Such a clause ensures that (at least) one of the n vertices is the i th vertex in the vertex cover.
- $\forall m \in [1, n], \forall p, q \in [1, k]$ with $p < q$, a clause $\bar{x}_{m,p} \vee \bar{x}_{m,q}$. Such a clause ensures that the vertex m cannot be both the p th and q th vertex in the vertex cover, for $p \neq q$.
- $\forall m \in [1, k], \forall p, q \in [1, n]$ with $p < q$, a clause $\bar{x}_{p,m} \vee \bar{x}_{q,m}$. Such a clause ensures that the vertex m in the vertex cover cannot be mapped into both p and q for $p \neq q$.
- For each edge $(i, j) \in E$, a clause $x_{i,1} \vee x_{i,2} \vee \dots \vee x_{i,k} \vee x_{j,1} \vee x_{j,2} \vee \dots \vee x_{j,k}$. Such a clause ensures that at least one of i and j is in the vertex cover.

► 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**

یکسری شرایط رو باید روی این متغیرهای باینری اعمال کنیم که اون شرایطی که مسئله vertex cover می خواد تامین بشه

1- باید بردار خروجی که k تا عضو و k تا راس توشه

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

- Competitions to find efficient SAT solvers



The international SAT Competitions web page

Current Competition

SAT 2019 Race

Organizers [Marlin Heule, Matti Järvisalo, Martin Suda](#)

Past Competitions

SAT 2018 Competition

Organizers [Marlin Heule, Matti Järvisalo, Martin Suda](#)

SAT 2017 Competition

Organizers [Marlin 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	YatSAT	lch glucose3	Score2SAT
	Parallel Track			No-Limit Track			Incremental Library Track		
SAT+UNSAT	Syrup24, Syrup48	Plingeling	Painless MapleCOMSPS	COMiniSATPS Pulsar	MapleCOMSPS LRB VSIDS 2, MapleCOMSPS LRB VSIDS	CaDiCaL NoProof	AbcdSAT	Glucose	Riss

SAT 2016 Competition

Organizers [Marlin 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).

مسئله NP به مسئله NPC

TSP به عنوان یک مسئله NPC مرجع در نظر گرفته شده و می‌خوایم چرخه هامیلتونی رو که گفتیم NP

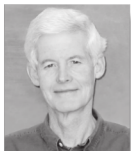
هست به TSP تبدیل کنیم

توی این چرخه هامیلتونی یال‌ها وزنی ندارند و خروجی بله است اگر یک‌تور از همه رئوس یکبار عبور کرده باشه در غیر اینصورت خروجی میشه خیر

Proof of the reduction correctness:

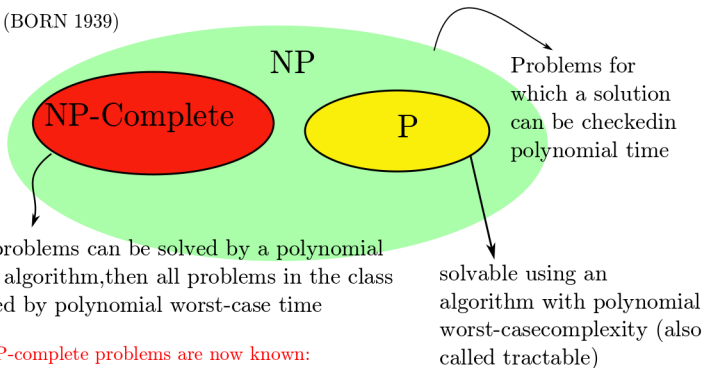
If the graph G has a Hamiltonian cycle, then this exact same tour will correspond to n edges in E' , each with weight 1. This gives a TSP tour in G' of weight exactly n . If G does not have a Hamiltonian cycle, then there can be no such TSP tour in G' because the only way to get a tour of cost n in G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

Summery



STEPHEN COOK (BORN 1939)

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.



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

