# COMPUTABILITY AND COMPUTATIONAL COMPLEXITY

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

# Agenda

# Introduction[1]

Tractable vs. intractable problems

- Tractable: problems that can be solved in polynomial time (algorithms in $O(p(n))$, where $p(n)$ is a polynomial in terms of $n$)
- Intractable: problems that cannot be solved in polynomial time

| $n$ | $\log_2 n$ | $n$ | $n \log_2 n$ | $n^2$ | $n^3$ | $2^n$ | $n!$ |
|---|---|---|---|---|---|---|---|
| 10 | 3.3 | $10^1$ | $3.3 \cdot 10^1$ | $10^2$ | $10^3$ | $10^3$ | $3.6 \cdot 10^6$ |
| $10^2$ | 6.6 | $10^2$ | $6.6 \cdot 10^2$ | $10^4$ | $10^6$ | $1.3 \cdot 10^{30}$ | $9.3 \cdot 10^{157}$ |
| $10^3$ | 10 | $10^3$ | $1.0 \cdot 10^4$ | $10^6$ | $10^9$ | | |
| $10^4$ | 13 | $10^4$ | $1.3 \cdot 10^5$ | $10^8$ | $10^{12}$ | | |
| $10^5$ | 17 | $10^5$ | $1.7 \cdot 10^6$ | $10^{10}$ | $10^{15}$ | | |
| $10^6$ | 20 | $10^6$ | $2.0 \cdot 10^7$ | $10^{12}$ | $10^{18}$ | | |

Performing a trillion ($10^{12}$) operations per second

- $2^{100}$ operations would take $4 * 10^{10}$ years
- Estimated age of Earth: $4.5 * 10^9$ years

Centro de Informática UFPE

Universidade Federal de Pernambuco

[1] Source: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

# Introduction

Complexity classes: P and NP

P: the class of decision problems that can be solved in polynomial time by deterministic algorithms

- Decision problems: with yes/no answers

Many important problems are not decision problems, but they can be reduced to a series of decision problems

- Original: what is the shortest path between $u$ and $w$?
- Reduction
  - Does a path of weight $\leq d$ exist?
  - Does a path of weight $\leq d - 1$ exist?
  - ...

# Introduction

There are many important problems for which
no polynomial-time algorithm has been found so far

- TSP – travelling salesman problem
- Subset sum problem
- Knapsack problem
- Graph coloring problem

Although solving can be difficult, checking
a candidate solution can be easy (tractable)

# Agenda



Centro de Informática UFPE

Universidade Federal de Pernambuco

# Class NP

Non-deterministic algorithms: two-stage procedure

- Non-deterministic: generating an arbitrary candidate solution
- Deterministic: checking whether it is a solution indeed

A non-deterministic algorithm solves a decision problem if it is capable of guessing a solution at least once and to be able to verify its validity

A non-deterministic algorithm is said to be non-deterministic polynomial if the time efficiency of its verification stage is polynomial

# Class NP

Hypothetical non-deterministic programming language

- All typical commands of a programming language in addition to a non-deterministic jump (nd-jump)

Non-deterministic algorithm

- Yes: if there is at least one way of performing non-deterministic jumps in order to find a valid solution
- No: otherwise

Example: *k*-clique problem

- Let $G = (V, E)$ be an undirected graph with no weights, $k \in \mathbb{N}$ such that $k \leq \mid V \mid$, is there a complete subgraph of $G$ with $k$ vertices?

# Class NP

**Algorithm:** bool clique(Graph G, int k)

```
for i ← 0 to n(G) − 1 do
    nd-jump{chosen[i] ← true | chosen[i] ← false};
for v ← 0 to n(G) − 1 do
    if chosen[v] then
        cont ← 0;
        w ← first(G, v);
        while w < n(G) do
            if chosen[w] then cont++;
            w ← next(G, v, w);
        if cont ≠ k − 1 then return false;
return true;
```

Time efficiency

- With nd-jump:
  $\Theta(\mid V \mid) + \Theta(\mid V \mid + \mid E \mid) = \Theta(\mid V \mid^2)$
- With no nd-jump:
  $\Theta(2^{\mid V \mid}) + \Theta(\mid V \mid + \mid E \mid) = \Theta(2^{\mid V \mid})$

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Class NP

NP: the class of decision problems that can be solved in polynomial time by non-deterministic algorithms

- $P \subseteq NP$

The most important open question of theoretical computer science

$$P \stackrel{?}{=} NP$$

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Agenda



Centro de Informática UFPE

Universidade Federal de Pernambuco

# Class NP-complete

Informally, NP-complete: the most difficult problems in NP

- If $P \neq NP$, then
  - $P \subset NP$
  - $P \cap$ NP-complete = $\emptyset$
  - NP-intermediate = NP $-$ NP-complete $-$ P
- If P = NP, then
  - P = NP = NP-Complete
  - NP-intermediate = $\emptyset$

Centro de Informática UFPE
UNIVERSIDADE FEDERAL DE PERNAMBUCO

# Class NP-complete

A decision problem $D$ is said to be in NP-complete if:

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

A decision problem $D_1$ is said to be polynomially reducible to a decision problem $D_2$, if there exists a function $t$ that transforms instances of $D_1$ to instances of $D_2$ such that:

- For all instances $i \in D_1$, the answer of $i$ is yes if, and only if, the answer of $t(i) \in D_2$ is yes
- $t$ is computable by a polynomial time algorithm

# Class NP-complete

Showing that a decision problem $D$ is NP-complete

1. $D$ is in NP: a candidate solution can be checked in polynomial time
2. Every problem in NP is reducible to $D$ in polynomial time

Due to the transitivity of polynomial reduction

2. Show that a known NP-complete problem can be reduced to $D$ in polynomial time

Cook-Levin's theorem

- The boolean satisfiability problem (SAT) is NP-complete

Richard Karp's theorem

- The 3-SAT problem is NP-complete

Centro de Informática UFPE

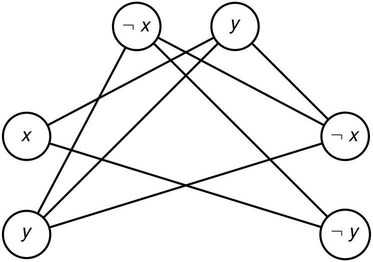
Universidade Federal de Pernambuco

# Class NP-complete

3-SAT: let $\alpha$ be a propositional logic formula (boolean expression) in conjunctive normal formal (CNF) with at most three literals per clause, is there $\alpha$ satisfiable?

Polynomial reduction of 3-SAT to k-clique, where $k = \#\text{clauses in } \alpha$

- Let $\alpha = c_1 \wedge c_2 \wedge \ldots \wedge c_m$, create the graph $G = (V, E)$ such that
  - $V \leftarrow \{v_{i,j}\}$, where $i$ denotes a clause and $j$ denotes a literal
  - $E \leftarrow \{(v_{i,j}, v_{k,l}) \mid i \neq k \wedge v_{i,j} \neq \neg\, v_{k,l}\}$



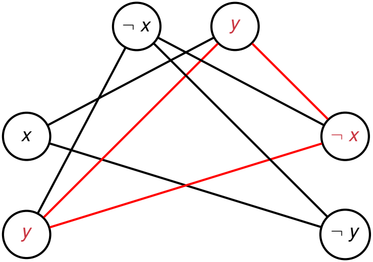

$$\begin{aligned}\alpha = {} & (x \vee y) \wedge \\ & (\neg\, x \vee y) \wedge \\ & (\neg\, x \vee \neg\, y)\end{aligned}$$

# Class NP-complete

3-SAT: let $\alpha$ be a propositional logic formula (boolean expression) in conjunctive normal formal (CNF) with at most three literals per clause, is there $\alpha$ satisfiable?

Polynomial reduction of 3-SAT to k-clique, where $k = \#$clauses in $\alpha$

- Let $\alpha = c_1 \wedge c_2 \wedge ... \wedge c_m$, create the graph $G = (V, E)$ such that
  - $V \leftarrow \{v_{i,j}\}$, where $i$ denotes a clause and $j$ denotes a literal
  - $E \leftarrow \{(v_{i,j}, v_{k,l}) \mid i \neq k \wedge v_{i,j} \neq \neg v_{k,l}\}$




$$\alpha = (x \vee y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

sol.: $x = false$, $y = true$

# Class NP-Complete

IF $3-SAT(\alpha) = yes \Rightarrow clique(t(\alpha), k) = yes$

- Since $3-SAT(\alpha) = yes$ and $\alpha$ is in CNF, there is at least one literal in each clause whose value is true. Considering the definition of $t$, the subgraph considering the corresponding nodes is a k-clique.

If $clique(t(\alpha), k) = yes \Rightarrow 3-SAT(\alpha) = yes$

- Considering the definition of $t$, each vertex corresponds to a literal and the edges link vertices whose corresponding literals belong to different clauses and they do not create a contradiction. Since $clique(t(\alpha), k) = yes$, it is possible to create a valuation based on the $k$ selected vertices that makes $\alpha$ satisfiable.

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Class NP-complete

Can a problem be more difficult to answer than those in NP-complete?

- Yes!

Informally, NP-hard: at least as hard as the hardest problems in NP

- NP-complete $\subseteq$ NP-hard

Are there other complexity classes?

- More than 500 complexity classes
- Complexity zoo: `https://complexityzoo.uwaterloo.ca/`

# Agenda

# Computability

There are problems: tractable and intractable, but they are decidable

There are undecidable problems (non-computable)

- Halting problem
- Entscheidungsproblem

# Computability

Suppose that there is an algorithm $A$ to solve the halting problem

- $A(P, I) = 1$, if $P$ halts on $I$
- $A(P, I) = 0$, if $P$ does not halt on $I$

Let $Q$ be defined as follows:

- $Q(P)$ halts if $A(P, P) = 0$
- $Q(P)$ does not halt if $A(P, P) = 1$

Substituting $P$ by $Q$ we reach a contradiction:

- $Q(Q)$ halts if $A(Q, Q)$ does not halt
- $Q(Q)$ does not halt if $A(Q, Q)$ halts

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Computability

How to address intractible and (even worse!) undecidable problems?

- Backtracking
- Branch and bound
- Dynamic programming
- Approximation algorithms
- Heuristics

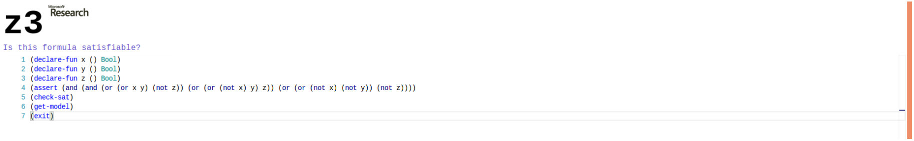
Centro de Informática UFPE

Universidade Federal de Pernambuco

# Computability[2]

Z3: an SMT solver developed by Microsoft

- 2015 – Prog. languages software award from ACM SIGPLAN
- 2018 – Test of time award from the ETAPS
- 2019 – Herbrand award for contributions to automated reasoning

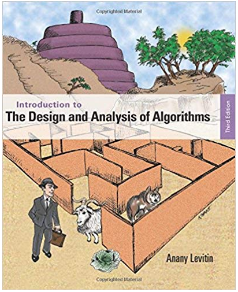



Example: `./z3 -smt2 sat_ex`

[2] Link: https://rise4fun.com/z3

# Agenda

# Bibliography




**Chapter 11 (pp. 401–409)**
**Anany Levitin.**
*Introduction to the Design and Analysis of Algorithms*.
3rd edition. Pearson. 2011.


**Chapter 17 (pp. 535–551, pp. 555–561)**
**Clifford Shaffer.**
*Data Structures and Algorithm Analysis*. Dover, 2013.

# COMPUTABILITY AND COMPUTATIONAL COMPLEXITY

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil