Computer modeling: computational complexity.

Jakub Tworzydło

Institute of Theoretical Physicis
Jakub.Tworzydlo@fuw.edu.pl

15/03/2022 Pasteura, Warszawa

Plan

Plan

Plan

Plan

Computational complexity

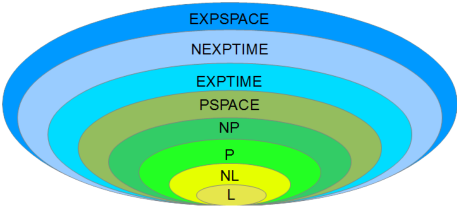
Today: surprising interconnections between computational complexity and statistical physics.

Computational complexity

Today: surprising interconnections between computational complexity and statistical physics.

Formal theory of computation: study how resources (e.g. CPU time, memory) scale with the size N of the problem.

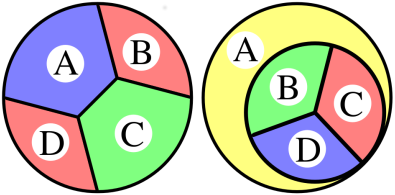
e.g. polynomial time algorithm ($t \sim N^2$) versus exponential time ($t \sim 2^N$)



Complexity Class	Description
L, LSPACE	Logarithmic space, deterministic
NL	Logarithmic space, non-deterministic
P, PTIME	Polynomial time, deterministic
NP	Polynomial time, non-deterministic
PSPACE	Polynomial space
EXP, EXPTIME	Exponential time, deterministic
NEXP, NEXPTIME	Exponential time, non-deterministic
EXPSPACE	Exponential space

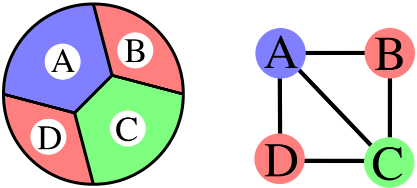
Two examples

Example 1
Can a map/graph containing N nodes be painted in three colors?



Two examples

Example 1
Can a map/graph containing N nodes be painted in three colors?



... find a good answer among many possibilities (3^N colorings)

Two examples

Example 2

Imagine we have to solve a constrain, which is expressed as a long logical expression (operators AND, OR, NOT) in N boolean variables:

$$(x_1 \vee x_2 \vee \neg x_3) \ \wedge \ (\neg x_1 \vee \neg x_2 \vee x_3) \ \wedge \ (\neg x_1 \vee \neg x_2 \vee \neg x_3).$$

Two examples

Example 2
Imagine we have to solve a constrain, which is expressed as a long logical expression (operators AND, OR, NOT) in N boolean variables:

$$(x_1 \vee x_2 \vee \neg x_3) \;\wedge\; (\neg x_1 \vee \neg x_2 \vee x_3) \;\wedge\; (\neg x_1 \vee \neg x_2 \vee \neg x_3).$$

Can we assign boolean values True/False to all the variables such, that the constrain is satisfied? This is a satisfiability problem or SAT.

Standardized form

CNF – Conjunctive Normal Form:
fixed, standard format for expressing logical formulas as a conjunction of clauses (alternatives)

$$(\mathrm{OR},\ \mathrm{NOT}) \wedge (\cdots\cdots) \wedge (\cdots\cdots)$$

kSAT – if every clause contains (at most) k variables

Mapping

We can rewrite 3-coloring problem in a SAT form:

A_R, A_G, A_B – variables for node A in red, blue and green

Mapping

We can rewrite 3-coloring problem in a SAT form:

A_R, A_G, A_B – variables for node A in red, blue and green

1) $A_R \vee A_G \vee A_B$ – condition that one of the colors is used
2) $\neg A_R \vee \neg A_G$ – the color is not red and green simultanously
3) $(\neg A_R \vee \neg B_R) \wedge (\neg A_G \vee \neg B_G) \wedge (\neg A_B \vee \neg B_B)$ – A and B vertices are not of the same color

Mapping

We can rewrite 3-coloring problem in a SAT form:

A_R, A_G, A_B – variables for node A in red, blue and green

1) $A_R \vee A_G \vee A_B$ – condition that one of the colors is used
2) $\neg A_R \vee \neg A_G$ – the color is not red and green simultanously
3) $(\neg A_R \vee \neg B_R) \wedge (\neg A_G \vee \neg B_G) \wedge (\neg A_B \vee \neg B_B)$ – A and B vertices are not of the same color

Three colors covering problem reduces to **3SAT** in $\mathcal{O}(N^2)$ time:
1) N clauses with 3 variables
2) $3N$ clauses with 2 variables
3) at most $3N(N-1)/2$ clauses with 2 variables (all possible edges)

Plan

Classes **P** and **NP**

What is important for us

- P class
- NP class
- NP-complete

Definitions are based on an abstract model of computer: so called Turing machine (deterministic sequential machine).

Classes **P** and **NP**

What is important for us

- P class
- NP class
- NP-complete

Definitions are based on an abstract model of computer: so called Turing machine (deterministic sequential machine).

P class

- set of problems solvable in polynomial time in N

NP class

- set of decision problems for which a solution can be checked in polynomial time in N
- **NP** means *non-deterministic polynomial*

P and **NP** properties, examples

P class

- “treatable”, solution on a computer is practical
- sorting, multipling, linear algebra, testing a prime number

P and **NP** properties, examples

P class

- “treatable”, solution on a computer is practical
- sorting, multipling, linear algebra, testing a prime number

NP class

- best known algorithms have “worst case scenario” complexity, which is exponential in N
- factoring a number
- we don’t have a proof that **P**$\neq$**NP**

P and **NP** properties, examples

P class

- “treatable”, solution on a computer is practical
- sorting, multipling, linear algebra, testing a prime number

NP class

- best known algorithms have “worst case scenario” complexity, which is exponential in N
- factoring a number
- we don’t have a proof that **P**$\neq$**NP**

NP-complete

- every other problem in **NP** is reducible to it in polynomial time, while the problem itself is **NP**
- can be used to solve any **NP** problem (maximally difficult)
- **kSAT** for $k \geq 3$, graph coloring, traveling salesman

Algorithm for **kSAT**

Davis-Putnam-Logemann-Loveland (DPLL):
search algorithm that decides satifiability of any **CNF**; recursive, backtracking algorithm with some simplifications (1962); still used as a basis for modern SAT-solvers

$(p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee r)$ set $p = 1 \longrightarrow$ reduces to $q \vee r$

Algorithm for **kSAT**

Davis-Putnam-Logemann-Loveland (DPLL):
search algorithm that decides satifiability of any **CNF**; recursive, backtracking algorithm with some simplifications (1962); still used as a basis for modern SAT-solvers

$(p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee r)$ set $p = 1 \longrightarrow$ reduces to $q \vee r$

DPLL routine: set a variable,
eliminate resolved clauses, call itself on reduced problem (recursion)
$\rightarrow$ either assignment or contradiction is found
$\rightarrow$ backtrack if contradiction is found

Algorithm for **kSAT**

Davis-Putnam-Logemann-Loveland (DPLL):
search algorithm that decides satifiability of any **CNF**; recursive, backtracking algorithm with some simplifications (1962); still used as a basis for modern SAT-solvers

$(p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee r)$ set $p = 1 \longrightarrow$ reduces to $q \vee r$

DPLL routine: set a variable,
eliminate resolved clauses, call itself on reduced problem (recursion)
- $\rightarrow$ either assignment or contradiction is found
- $\rightarrow$ backtrack if contradiction is found

DPLL simplification:
pure literal – first set variable, which shows up with the same “sign”
unit propagation – set variables in one-variable clauses

Plan

Spin glasses

model inspired by physics of magnetic materials:
N spins (magnetic moments) represented by $s_i = \pm 1$

the spins interact randomly, and may also experience (local) magnetic fields, system energy is

$$E(s_1, \ldots, s_N) = -\sum_{ij} J_{ij} s_i s_j - \sum_i h_i s_i$$

Spin glasses

model inspired by physics of magnetic materials:
N spins (magnetic moments) represented by $s_i = \pm 1$

the spins interact randomly, and may also experience (local) magnetic fields, system energy is

$$E(s_1, \ldots, s_N) = -\sum_{ij} J_{ij} s_i s_j - \sum_i h_i s_i$$

decision formulation: for a given energy E_0 is there a configuration $\{s_i\}_{i=1,\ldots,N}$ with smaller energy $E(\{s_i\}) < E_0$?

3D problem with random couplings J_{ij} was shown to be **NP**-complete: *geometric frustration* is the source of complexity

kSAT as spin-glass problem

Let's map a clause $(x_1 \vee \neg x_2 \vee x_3)$ to spin variables $s_i = 2x_i - 1$; note that $E = (1 - s_1)(1 + s_2)(1 - s_3)/2^3$ gives $E = 1$ when the clause is False ($x_1 = 0,\ x_2 = 1,\ x_3 = 0$) and $E = 0$ otherwise.

kSAT as spin-glass problem

Let's map a clause $(x_1 \vee \neg x_2 \vee x_3)$ to spin variables $s_i = 2x_i - 1$; note that $E = (1 - s_1)(1 + s_2)(1 - s_3)/2^3$ gives $E = 1$ when the clause is False ($x_1 = 0,\ x_2 = 1,\ x_3 = 0$) and $E = 0$ otherwise.

Total energy function measures number of unsatisfied clauses

$$E_{\mathrm{kSAT}} = \frac{1}{2^k} \sum_{j=1}^{M} \prod_{i=1}^{N} (1 - W_{ji} s_i),$$

where M number of clauses, N number of variables, table set to $W_{ji} = +1$ if clause j inludes variable i, $W_{ji} = -1$ if it includes negation and $W_{ji} = 0$ otherwise.

kSAT as spin-glass problem

Let's map a clause $(x_1 \vee \neg x_2 \vee x_3)$ to spin variables $s_i = 2x_i - 1$; note that $E = (1 - s_1)(1 + s_2)(1 - s_3)/2^3$ gives $E = 1$ when the clause is False ($x_1 = 0,\ x_2 = 1,\ x_3 = 0$) and $E = 0$ otherwise.

Total energy function measures number of unsatisfied clauses

$$E_{\mathrm{kSAT}} = \frac{1}{2^k} \sum_{j=1}^{M} \prod_{i=1}^{N} (1 - W_{ji} s_i),$$

where M number of clauses, N number of variables, table set to $W_{ji} = +1$ if clause j inludes variable i, $W_{ji} = -1$ if it includes negation and $W_{ji} = 0$ otherwise.

Remark: **2SAT** includes only "two-body" spin-spin interaciton; more on the explicit mappings of combinatorial problems see e.g. Andrew Lucas `https://arxiv.org/abs/1302.5843`

NP for physicists

Scott Aaronson (new Quantum Information Center at Austin):

postulates a fundamental impossibility to solve *NP*-complete problem in a physical process (including quantum mechanics)

Quantum Physics

NP-complete Problems and Physical Reality

Scott Aaronson

(Submitted on 12 Feb 2005 (v1), last revised 21 Feb 2005 (this version, v2))

Can NP-complete problems be solved efficiently in the physical universe? I survey proposals including soap bubbles, protein folding, quantum computing, quantum advice, quantum adiabatic algorithms, quantum-mechanical nonlinearities, hidden variables, relativistic time dilation, analog computing, Malament-Hogarth spacetimes, quantum gravity, closed timelike curves, and "anthropic computing." The section on soap bubbles even includes some "experimental" results. While I do not believe that any of the proposals will let us solve NP-complete problems efficiently, I argue that by studying them, we can learn something not only about computation but also about physics.

Comments:	23 pages, minor corrections
Subjects:	**Quantum Physics (quant-ph)**; Computational Complexity (cs.CC); General Relativity and Quantum Cosmology (gr-qc)
Journal reference:	ACM SIGACT News, March 2005
Cite as:	**arXiv:quant-ph/0502072**

Connection to spin-glasses

Mezard, Parisi, Zacchina (2002) proposed an algorithm inspired by simulated annealing: introducing temperature (as in statistical physics) to SAT problem.

It also become of interest to ask different questions on average satisifiability of random SAT problems, typical running time of solvers, and phase transitions.

Connection to spin-glasses

Mezard, Parisi, Zacchina (2002) proposed an algorithm inspired by simulated annealing: introducing temperature (as in statistical physics) to SAT problem.

It also become of interest to ask different questions on average satisifiability of random SAT problems, typical running time of solvers, and phase transitions.

Reading material should expand on this modern concepts, please read "Satisfied with physics" Since comment, and Kirkpatrick & Selman review paper (paragraphs 1.1-1.4).