👋 Welcome!

CIS 1921: Solving Hard Problems in Practice

Teaching Staff

ISHAAN LAL
Instructor
ilal @ seas

CINDY YANG
Teaching Assistant
cindyy @ seas

THOMAS NGULUBE
Teaching Assistant
tngulube @ seas

Data collection…

seas.upenn.edu/~cis192
1

COURSE WEBSITE

Goals for the course

- **Encounter** vital but provably hard problems
- **Discover** how industry experts tackle problems in practice
- **Experiment** with industrial tools using modern techniques
- **Apply** these tools to your own hard problems

Goals for the course

- **Encounter** vital but provably hard problems
- **Discover** how industry experts tackle problems in practice
- **Experiment** with industrial tools using modern techniques
- **Apply** these tools to your own hard problems

Goals for the course

- **Encounter** vital but provably hard problems
- **Discover** how industry experts tackle problems in practice
- **Experiment** with industrial tools using modern techniques
- **Apply** these tools to your own hard problems

Goals for the course

- **Encounter** vital but provably hard problems
- **Discover** how industry experts tackle problems in practice
- **Experiment** with industrial tools using modern techniques
- **Apply** these tools to your own hard problems

Course Logistics

- **Homework**: 5 programming assignments
  - Late policy: 48 hours cumulative late submission.
  - HW0 (finger exercises) due **Monday, January 27th, 11:59PM**.
- **Final Project**: solve your own hard problem, flexible
- **Exams**: no
- **Office Hours**: schedule (TBD) on seas.upenn.edu/~cis1921
- **Gradescope: 8KPRG5**
- **Canvas / Ed:** Working on it...
- **Prerequisites:** CIS1210 (strict), CIS2620 (nice to have, not necessary)

Grading

- Homework: 44%
- Final Project: 38%
- Quizzes: 10%
- Attendance: 8%

- Final grades:
  don't worry too much about it.

Academic Integrity

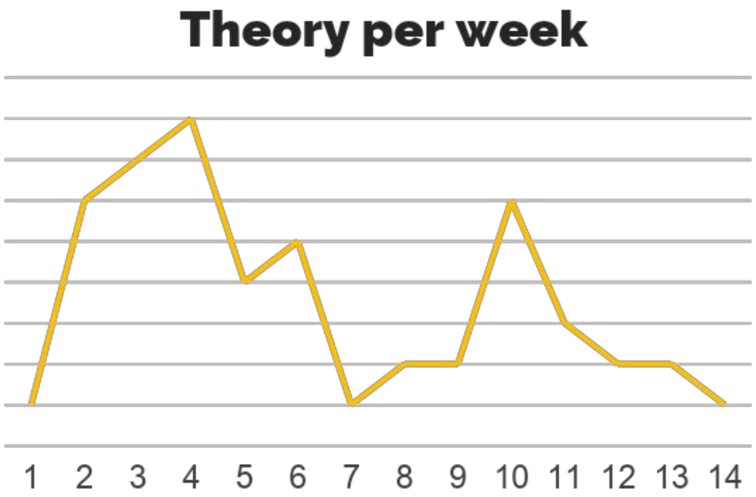
- Work on assignments individually (except final project)
  - Discussion encouraged, but work should be yours
- OK: high-level discussions
  - "Can you help me understand the DPLL algorithm?"
- OK: low-level discussions
  - "How do I time my program in OR-Tools?"
- Be careful: mid-level discussions
  - Not OK: "How exactly do I write this constraint?"

Health Logistics

- If you have a reasonable suspicion that you have Covid or sickness, **don't come**
  - Email me **before class** and we'll work something out

Theory? Practice?

- Interspersed theory & practice
  - “Practice”: applying techniques
  - “Theory”: how techniques work
  - Proofs rarely
- Lots of problem solving!
- Some “live coding demos”



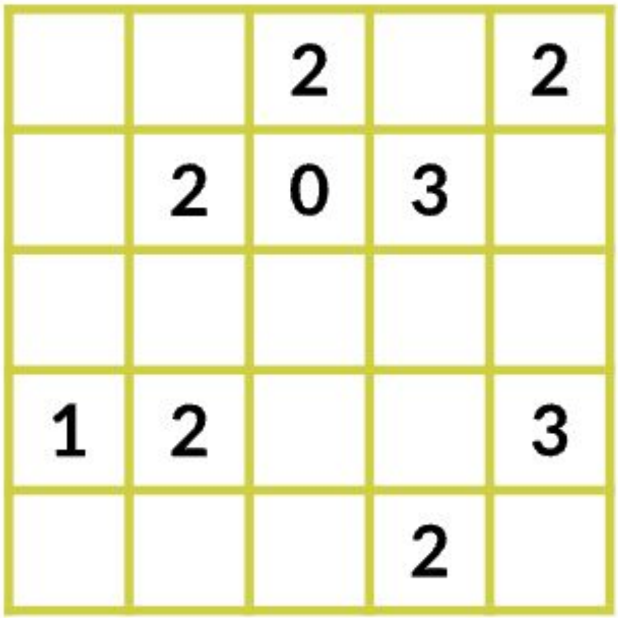
This graphic may be outdated...

Problem

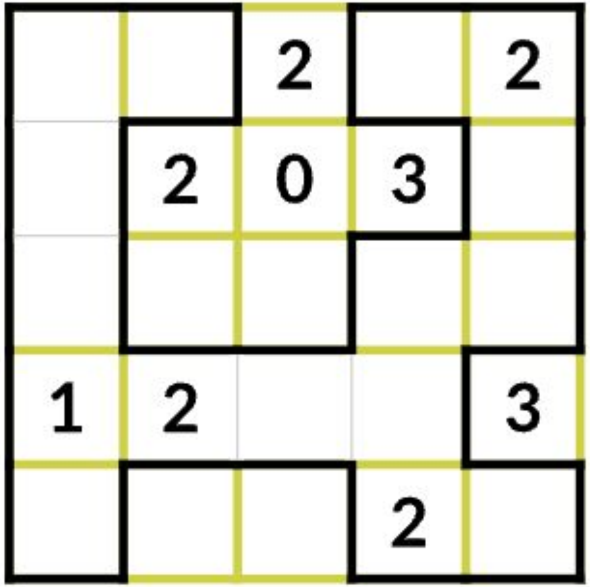
		2		2
	2	0	3	
1	2			3
			2	

Problem

- Draw lines along the edges of the squares to form a single loop without crosses or branches
- The number indicates how many lines surround it



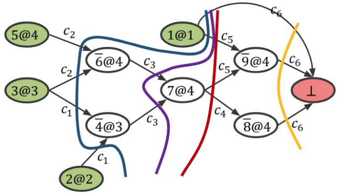
Solution

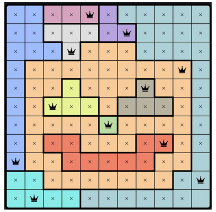


https://shorturl.at/6P4oW

lowercase letter “oh”

Timeline of the Semester





SAT, Graph Coloring, Matchings

Lec 1-2

Lec 3-4

Algorithms for SAT

LP, ILP, MP

Lec 5-6

Lec 8-9

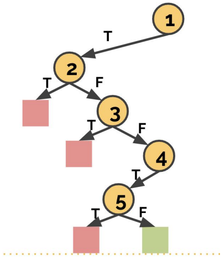
Constraint Programming

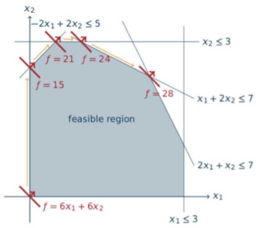
FDP, TSP, VRP

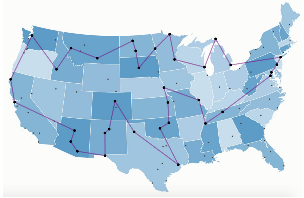
Lec 10-11

Lec 12-13

Special Topics, Heuristics, Genetic Algorithms







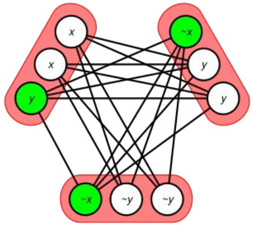
Timeline of Homeworks

HW 0

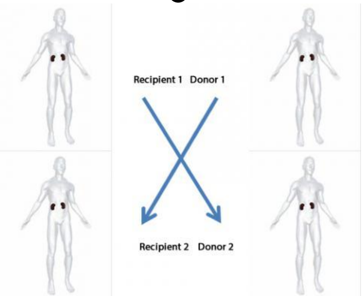
HW 1

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

HW 2



HW 3



HW 4

Project

Class Schedule

- 7:00 - 7:10 – Quiz or TA-led review
- 7:10 - 8:30 – Lecture
  - maybe end sooner...

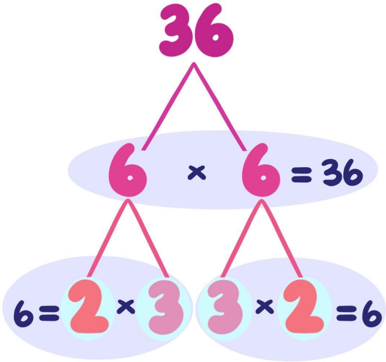
CIS 1921

Lecture 1: Hard Problems

What makes a problem **hard?**

CIS 262 in 5 minutes

- **DECISION problem**: some question that can be answered YES/NO for any input



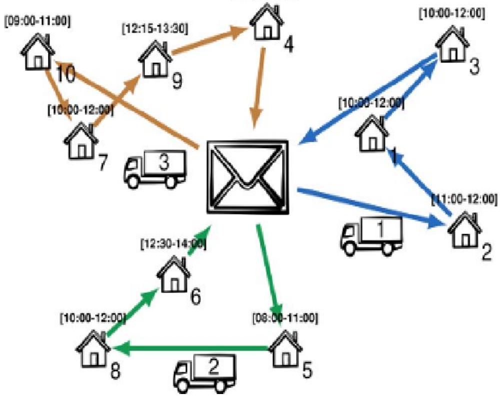
- **OPTIMIZATION problem**: try to find the "best" out of many feasible solutions

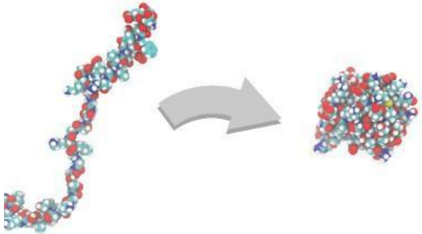
CIS 262 in 5 minutes

- Easy problem: we can solve it quickly for *any* input
  - Quickly: as input size grows, solving time grows at most polynomially
- Difficult problem: can't solve it quickly for every input
  - Solving time might grow exponentially in general

CIS 262 in 5 minutes

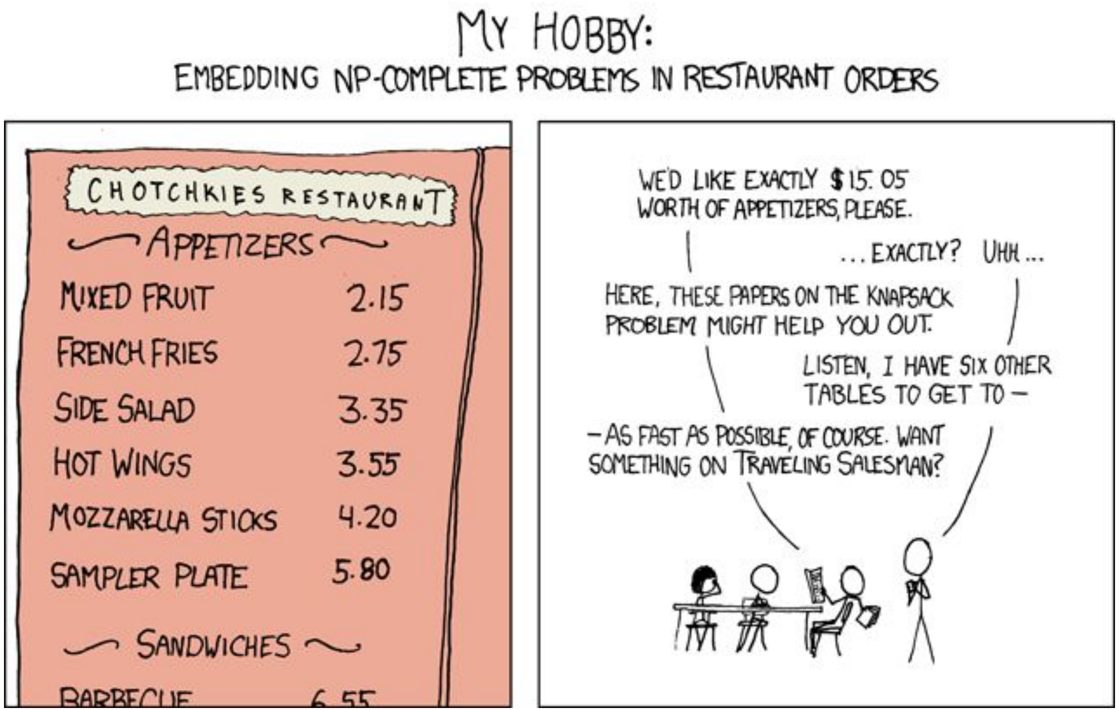
- NP-complete: tons of critical decision problems that turn out to be equivalent





protein folding

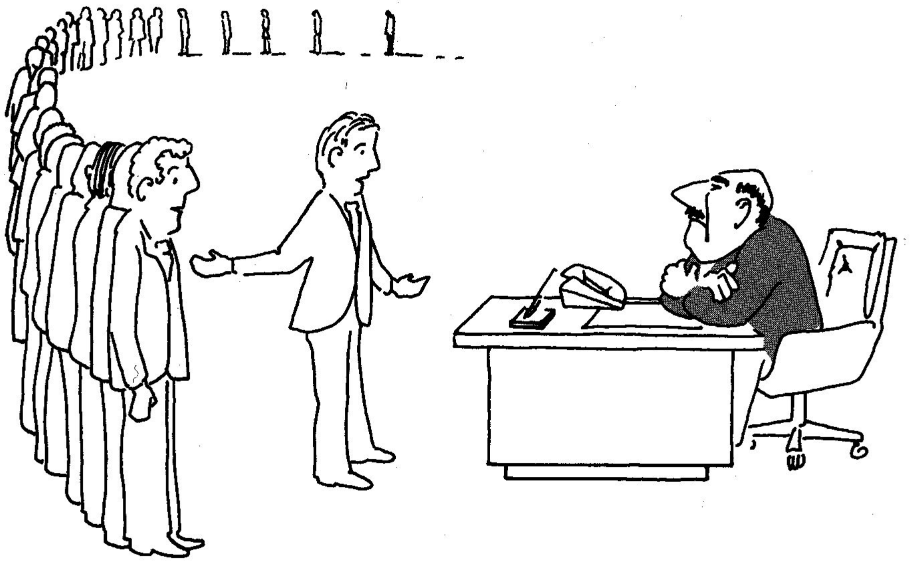
CIS 262 in 5 minutes



CIS 262 in 5 minutes

- Probably difficult: nobody has able to figure out how to solve these problems quickly in 50+ years

Our final definition of hard



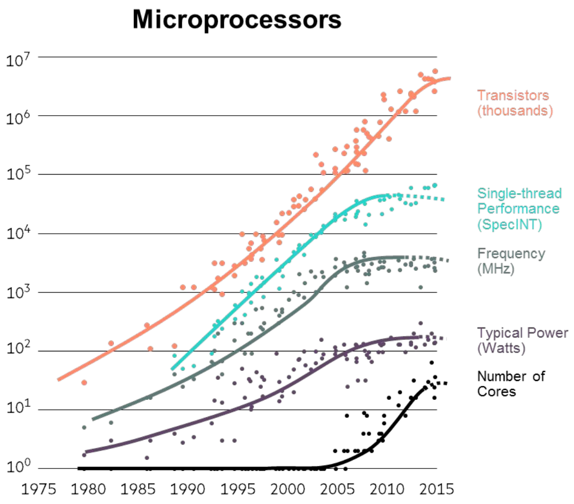
“I can’t find an efficient algorithm, but neither can all these famous people.”

We'll look at NP-complete problems (both decision and optimization varieties) in this course.

- Decision problems often ask "does there exist some solution?"
- In practice, we don't just want to determine if a solution exists; want to **find** a solution as well.

Does exponential runtime matter?

- **Moore, 1965:** number of transistors per chip **doubles** every two years
- Why bother with solvers? Just wait for faster computers
  - **Issue 1:** if problems take $O(2^n)$ time, then even if computer speed doubles, we can only increase n by 1
  - **Issue 2:** 55 years later, Moore's law is slowing down

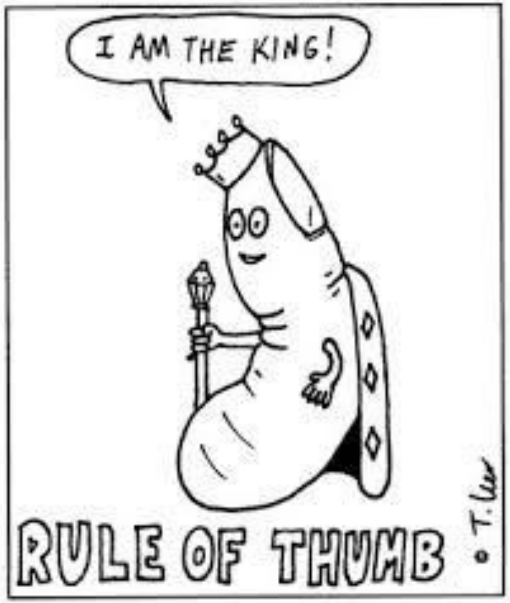


How to solve it, then?

A hopeless challenge?

No! Worst case is pessimistic!

Heuristic Algorithms

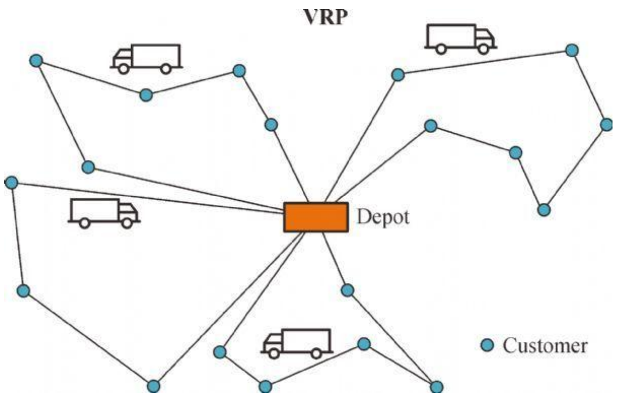


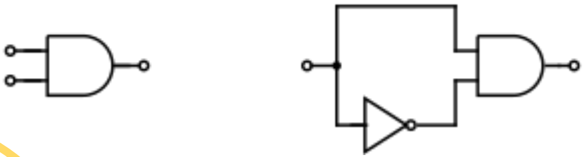
The Universal Solver

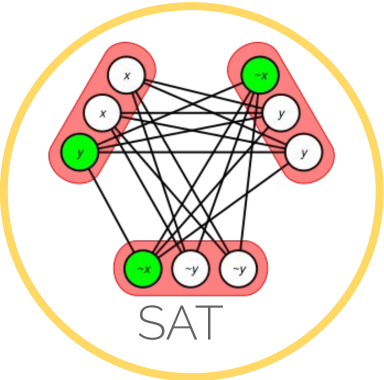
- 1956-74: early efforts towards **general automated reasoning**
  - **1956:** Samuel's checkers program demonstrated on TV
  - **1959:** Simon, Shaw & Newell's *General Problem Solver*
  - **1964:** Bobrow's natural-language word problem solver
- 1971: introduction of NP-completeness
  - **General idea:** can solve one problem extremely well, and reduce all other problems to that problem

Classic hard problem: SAT

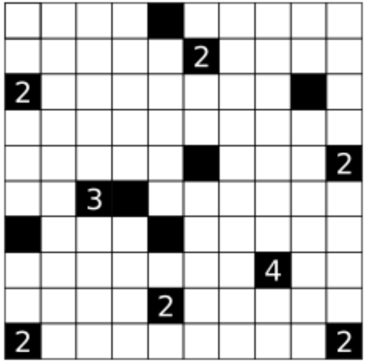
- **Satisfiability Problem:** Given a formula of boolean variables, does there exist a truth assignment that makes the entire formula evaluate to True?
  - Is this a **decision** problem or an **optimization** problem?
  - Many problems can be encoded as SAT instances
  - Assignment: a choice of truth values for each variable.
- **Let's see some examples on the board...**
- **Cook's Theorem (1971): SAT is NP-complete**
  - First NP-complete problem!

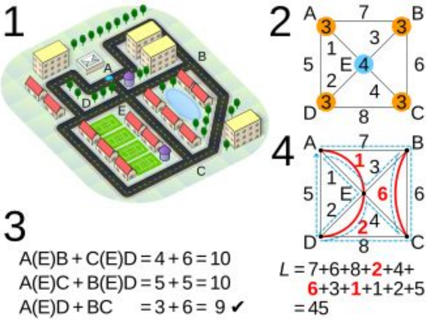

VRP
Depot
Customer




x
x
y
~x
y
y
~x
~y
~y
SAT


$4
12kg
?
15 kg
$2
2kg
$10
4kg


2
2
2
3
4
2
2
2


1
2
3
4
A
B
C
D
E
7
3
5
1
2
4
6
8
A(E)B + C(E)D = 4 + 6 = 10
A(E)C + B(E)D = 5 + 5 = 10
A(E)D + BC = 3 + 6 = 9 ✔
L = 7+6+8+2+4+
6+3+1+1+2+5
= 45

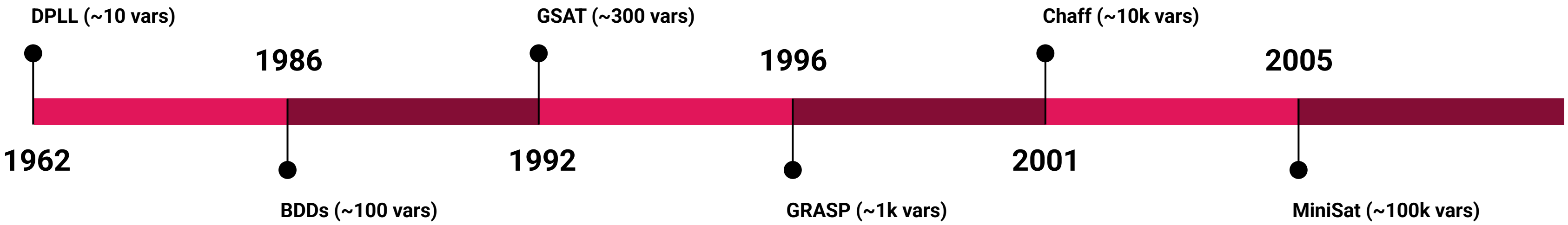

Clifton
Newark
New York
Huntington
Long Beach

Modern SAT solvers

- **SAT solvers:** black-boxes to quickly solve huge instances of SAT
- 1962: Davis, Putnam, and Loveland formulate precursor to most modern SAT solvers
- **GRASP** (UMich 1996) and **Chaff** (Princeton 2001): first practical, efficient SAT solvers

- The improvement in the performance of SAT solvers over the past 20 years is *revolutionary*!
  - Better marketing: Deep Solving

Timeline of SAT solvers



- Today: can solve instances with **millions** of variables
  - 1m vars: search space of assignments is $2^{1000000} \approx 9.9 \times 10^{301029}$
  - Age of universe $\approx 4.3 \times 10^{26}$ nanoseconds
- This chart refers to typical SAT instances found in industry applications

SAT terminology

- Assume only logical symbols are AND, OR, NOT
- **Literal:** a boolean variable (x) or its negation ($\overline{x}$)
  - (x) is called a **positive** literal, and ($\overline{x}$) is a **negative** literal
  - "a variable as it appears in a formula"
- **Clause:** a disjunction/OR of literals
  - e.g. $(\overline{x} \vee y \vee z)$

- Let's see an **example...**

Conjunctive Normal Form

- A boolean formula is in **conjunctive normal form (CNF)** if it is a conjunction/AND of clauses (i.e., an AND of ORs)
  - "a CNF" means "a formula in CNF"
- Ex: which of the following are in CNF?
  - $(\overline{x} \vee y \vee z) \wedge (x \Rightarrow w)$
  - $(\overline{x} \wedge y \wedge z) \vee (\overline{y} \wedge z)$
  - $(\overline{x} \vee y \vee z) \wedge (\overline{y} \vee z)$
  - $\overline{x} \vee y \vee z$
  - $x \wedge \overline{x}$

CNF-SAT: a loss of generality?

- It's convenient for SAT solvers to accept formulas in CNF, but what if we need to solve any other non-CNF boolean formula?
- **Every boolean formula φ can be expressed in CNF**
  - Rewrite in terms of ∧,∨, ¬
  - Apply distributive & DeMorgan's laws until formula is in CNF

DeMorgan's & Distributive Laws

Important Logical Properties

$\neg(p \lor q) \equiv \neg p \land \neg q$	(DeMorgan's Law)
$\neg(p \land q) \equiv \neg p \lor \neg q$	(DeMorgan's Law)
$(p \lor q) \lor r \equiv p \lor q \lor r$	(Associativity of $\lor$)
$(p \land q) \land r \equiv p \land q \land r$	(Associativity of $\land$)
$(p \land q) \lor r \equiv (p \lor r) \land (q \lor r)$	(Distributive property of $\lor$)
$(p \lor q) \land r \equiv (p \land r) \lor (q \land r)$	(Distributive property of $\land$)

Next week: learn how to use SAT solvers ourselves!

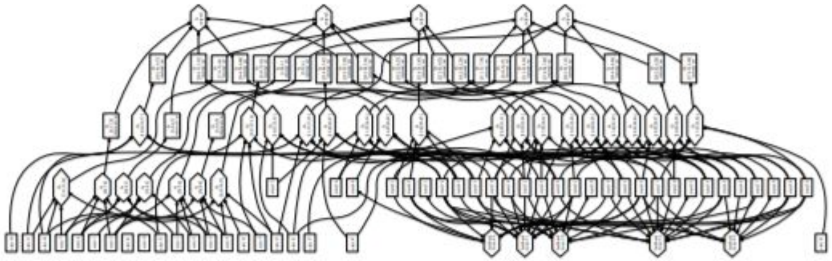


Fig. 4. Clause- and variable-dependency graph of HiTag2. Clause groups are represented as hexagons, and variables as boxes. The known keystream bits are the 5 final filter functions at the top, and the feedback functions are the 5 hexagons at the bottom right.

7.2.2.2 SATISFIABILITY: ONE HUNDRED TEST CASES 117

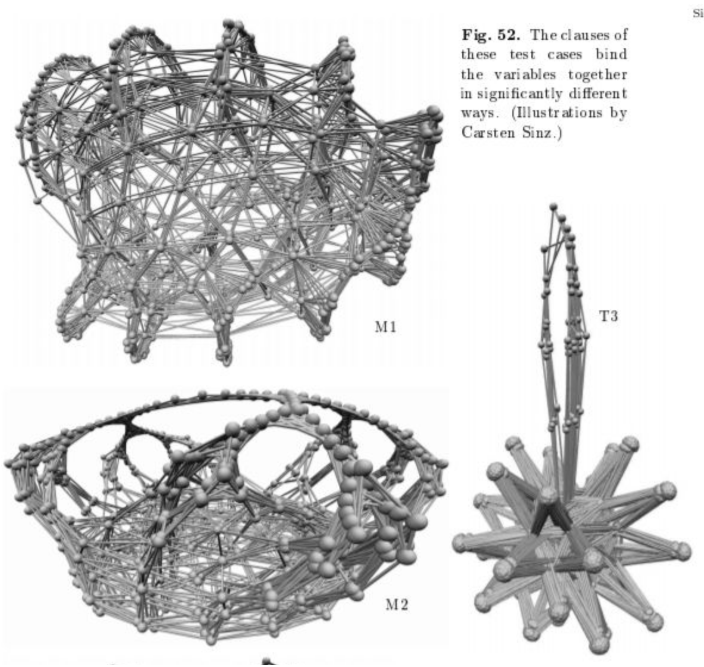


Fig. 52. The clauses of these test cases bind the variables together in significantly different ways. (Illustrations by Carsten Sinz.)

Our language of choice...

Python!

- Pros:
  - Easy to learn and use
  - Concise
  - Don't need to spend time worrying about low-level details
- Cons:
  - Slow (in practice, C++ is used to develop solver systems)

But I don’t know Python...

Don’t worry!

- HW0: Finger Exercises will bring you up to speed
- Very easy syntax, low learning curve
- Don’t need to be a Python expert to succeed in 1921
- If you are comfortable with any OOP language (e.g. Java) you’ll be fine