COMPSCI 311: Introduction to Algorithms

Lecture 20: Intractability: More Reductions, NP

Marius Minea

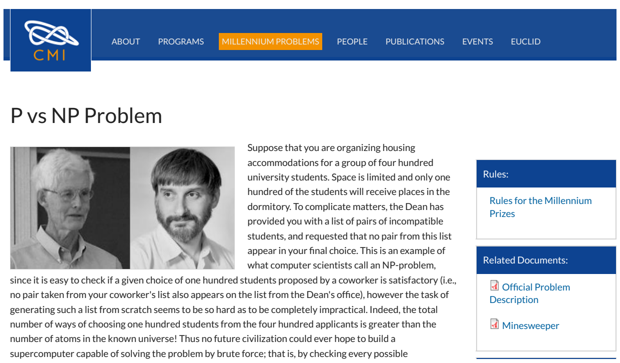
University of Massachusetts Amherst

slides credit: Dan Sheldon (adapted)

Reduction Strategies

- Reduction by equivalence (Vertex Cover and Independent Set)
- Reduction to a more general case (Vertex Cover to Set Cover)
- Reduction by "gadgets": Satisfiability

P=NP, $1M, and Minesweeper ?



Source: Clay Mathematics Institute, claymath.org

What does Minesweeper have to do with this ??

More Reduction: Satisfiability

- Can we determine if a Boolean formula has a satisfying assignment?

$$(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3)$$

- Boolean variables: $x_1, \ldots, x_n$
- Term: a variable or its negation. x_i or $\bar{x}_i$
- Clause: a disjunction ("or") of terms. $C = x_1 \vee \bar{x}_2 \vee x_4$
- Formula: a conjunction ("and") of clauses. $C_1 \wedge C_2 \wedge \ldots \wedge C_k$
- Assignment: assign 0/1 to each variable.
  $x_1 = 1, x_2 = 1, x_3 = 1$
- Satisfying assignment: makes all clauses evaluate to "true".
  $x_1 = 0, x_2 = 0, x_3 = 0$

Solving Satisfiability

SAT – Given boolean formula $C_1 \wedge C_2 \ldots \wedge C_m$ over variables $x_1, \ldots, x_n$, does there exist a satisfying assignment?

$$(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x}_3)$$

Some simplification rules:

If x_i is unit clause: $x_i \wedge (\ldots) \wedge (\ldots)$, x_i must be true
If $\bar{x}_j$ is unit clause, x_j must be false

Propagation: if x_i is true, all clauses with x_i are true
and $\bar{x}_i$ can be removed from all clauses

But, it no more simplifications, must still try both cases for x_i
worst-case asymptotics still exponential (brute force is 2^n)

SETH: Strong Exponential Time Hypothesis (more than P $\neq$ NP):
SAT cannot be solved in subexponential time in the worst case.

Back to Minesweeper

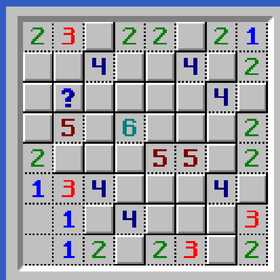
Playing Minesweeper well means not taking needless risks

$\Rightarrow$ reasoning about where mines may be

$\Rightarrow$ solving Boolean constraints

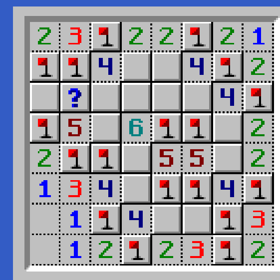
Richard Kaye, *Minesweeper is NP-complete!*, Mathematical Intelligencer, 2000

Playing the game



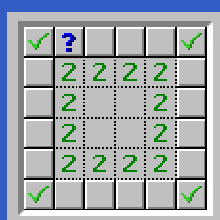
Does (2,6) have a mine?

Playing the game



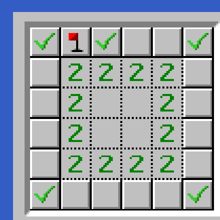
These must have mines

A puzzle



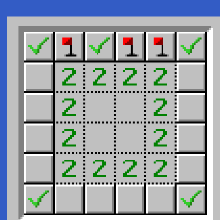
If this is a mine the one next to it is clear. . .

A puzzle



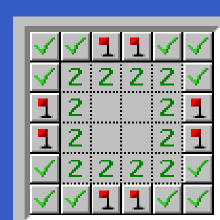
So. . .

A puzzle



. . . which is impossible!

A puzzle



Solved!

Minesweeper and SAT

To play Minesweeper, one must solve SAT problems

MINESWEEP $\leq_P$ SAT

Does this mean Minesweeper is NP-Complete?

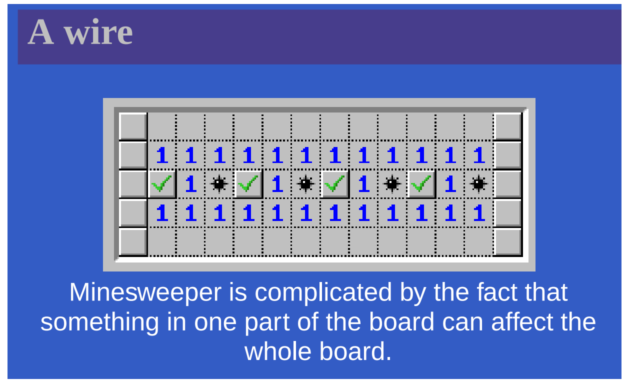
No! Every algorithmic problem can be expressed with Booleans

We need to reduce SAT to Minesweeper!

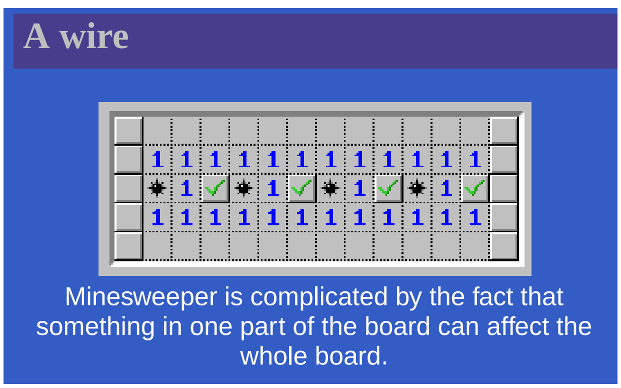
Can do it with a Boolean circuit version {CIRCUIT-SAT}

Encoding Circuits with Minesweeper

Simplest circuit: a wire (propagates a value)



Encoding Circuits with Minesweeper



(values of cells have shifted)

Can do (more complicated): AND, OR, XOR, wire crossings, splits

Reduction by Gadgets: Satisfiability

SAT – Given boolean formula $C_1 \wedge C_2 \ldots \wedge C_m$ over variables $x_1, \ldots, x_n$, does there exist a satisfying assignment?

3-SAT – Same, but each C_i has exactly three terms

Claim: 3-SAT $\leq_P$ INDEPENDENTSET.

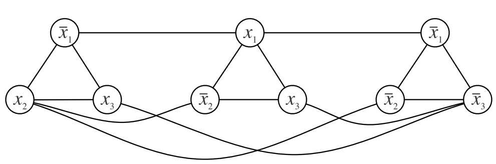
Reduction:

- Given 3-SAT instance $\Phi = \langle C_1, \ldots, C_m \rangle$, we will construct an independent set instance $\langle G, m \rangle$ such that G has an independent set of size m iff Φ is satisfiable
- Return YES if `solveIS`($\langle G, m \rangle$) = YES

Reduction

- **Idea**: construct graph G where independent set will select one term per clause to be true

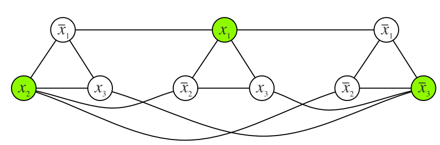
$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$



- One node per term
- Edges between all terms in same clause (select at most one)
- Edges between a literal and all of its negations (consistent truth assignment)

Correctness

$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

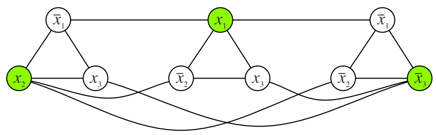


Claim: if G has an independent set of size m, then $\langle C_1, \ldots, C_m \rangle$ is satisfiable

- Suppose S is an independent set of size m
- Assign variables so selected literals are true. Edges from terms to negations ensure non-conflicting assignment.
- Set any remaining variables arbitrarily
- At most one term per clause is selected. Since m are selected, every clause is satisfied.

Correctness

$$(\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

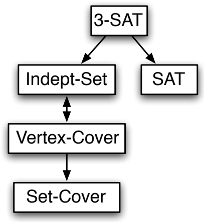


Claim: if $\langle C_1, \ldots, C_m \rangle$ is satisfiable, then G has an independent set of size m

- Consider any satsifying assignment of $\langle C_1, \ldots, C_m \rangle$
- Let S consist of one node per triangle corresponding to true literal in that clause. Then $|S| = m$.
- For (u, v) within clause, at most one endpoint is selected
- For edge $(x_i, \bar{x}_i)$ between clauses, at most one endpoint is selected, because $x_i = 1$ or $\bar{x}_i = 1$, but not both
- Therefore S is an independent set

Reductions So Far

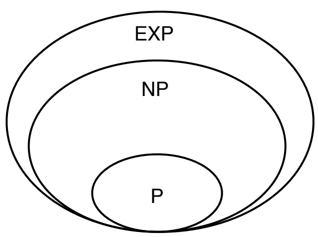
Partial map of problems we can use to solve others in polynomial time, through transitivity of reductions:



- $\boxed{Y} \rightarrow \boxed{X}$ means $Y \leq_P X$.

Toward a Definition of NP

Remember our problem hierarchy:



Let's formally define NP.

Remember: exponential time means $O(2^{n^d})$ for some constant d.

P and NP

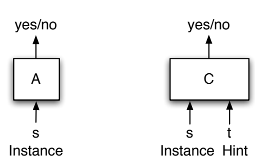
- P: Decision problems for which there is a polynomial time algorithm.
- NP: Decision problems for which there is a polynomial time certifier.

Intuition: A correct solution can be certified in polynomial time.

Solver vs. Certifier

Let X be a decision problem and s be problem instance (e.g., $s = \langle G, k \rangle$ for Independent Set)

Poly-time solver. Algorithm $A(s)$ such that $A(s) =$ Yes iff correct answer is Yes, and running time polynomial time in $|s|$



Poly-time certifier. Algorithm $C(s, t)$ such that for every instance s, there is some t such that $C(s, t) =$ Yes iff correct answer is Yes, and running time is polynomial in $|s|$.

- t is the "certificate" or hint. Must also be polynomial-size in $|s|$

Certifier Example: Independent Set

Input $s = \langle G, k \rangle$.
Problem: Does G have an independent set of size at least k?
Idea: Certificate $t =$ an independent set of size k

`CertifyIS`($\langle G, k \rangle, t$)
  if $|t| < k$ return No
  for each edge $e = (u, v) \in E$ **do**
    if $u \in t$ and $v \in t$ return No
  end for
  Return Yes

Polynomial time? Yes, linear in $|E|$.

Example: Independent Set

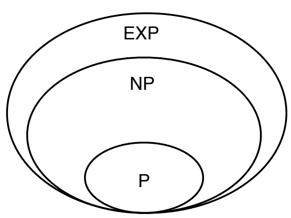
- Independent Set $\in$ P?
  - Unknown. No known polynomial time algorithm.
- Independent Set $\in$ NP?
  - Yes. Easy to certify solution in polynomial time.

Example: 3-SAT

Input: formula Φ on n variables.
Problem: Is Φ satisfiable?
Idea: Certificate t = the satisfying assignment

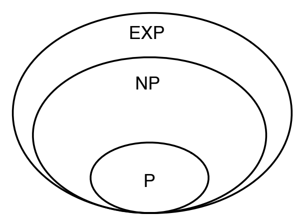
`Certify3SAT(` $\langle\Phi\rangle, t$`)`
 ▷ Check if t makes Φ true

Takeaway



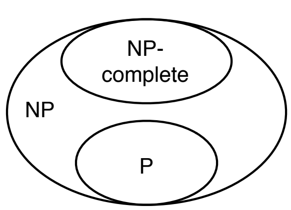
- 3SAT and Independent Set are in NP, as are many other problems that are hard to solve, but easy to certify!

P, NP, EXP



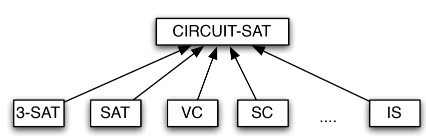
- **Claim**: P $\subseteq$ NP
- **Claim**: NP $\subseteq$ EXP
- Both straightforward to prove, but not critical right now.

NP-Complete



- NP-complete = a problem $Y \in$ NP with the property that $X \leq_P Y$ for every problem $X \in$ NP!

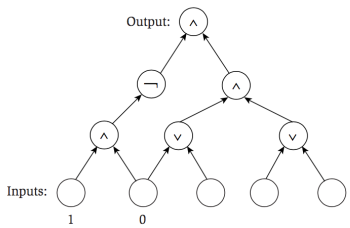
NP-Complete



- **Cook-Levin Theorem**: In 1971, Cook and Levin independently showed that particular problems were NP-Complete.
- We'll look at Circuit-SAT as canonical NP-Complete problem.

CIRCUIT-SAT

Problem: Given a circuit built of AND, OR, and NOT gates with some hard-coded inputs, is there a way to set remaining inputs so the output is 1?



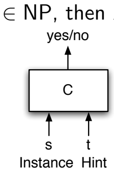
Satisfiable? Yes. Set inputs: 1, 1, 0.

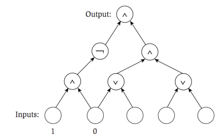
CIRCUIT-SAT

Cook-Levin Theorem CIRCUIT-SAT is NP-Complete.

Proof Idea: encode arbitrary certifier $C(s,t)$ as a circuit

- If $X \in$ NP, then X has a poly-time certifier $C(s,t)$

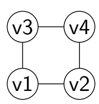


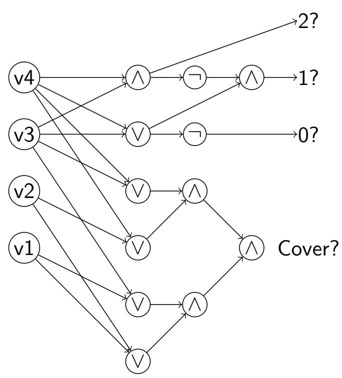


- Construct a circuit where s is hard-coded, and circuit is satisfiable iff $\exists t$ that causes $C(s,t)$ to output YES
- Algorithm for CIRCUIT-SAT implies an algorithm for X

A CIRCUIT-SAT reduction

- Vertex Cover – Does G have VC of size at most k?





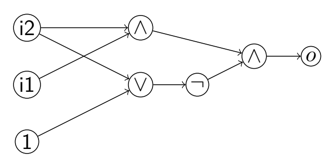
Back to 3-SAT

Claim: If Y is NP-complete and $Y \leq_P X$, then X is NP-complete.

Theorem: 3-SAT is NP-Complete.

- In NP? Yes, check satisfying assignment in poly-time.
- Prove by reduction from CIRCUIT-SAT.

Example.



Reduction: CIRCUIT-SAT $\leq_P$ 3-SAT

- One variable x_v per circuit node v plus clauses to enforce circuit computations
- Equality = equivalence (conjunction of two implications)
- An implication can be written as an "or" clause
  - $A \Rightarrow B$ is the same as $\neg A \vee B$
  - B can in turn be a disjunction
- Negation node: $x_v = \neg x_u$
  - $x_u \Rightarrow \neg x_v$
  - $\neg x_u \Rightarrow x_v$
- OR node: $x_v = x_u \vee x_w$
  - $x_u \Rightarrow x_v$
  - $x_w \Rightarrow x_v$
  - $x_v \Rightarrow x_u \vee x_w$
- AND node: $x_v = x_u \wedge x_w$
  - $x_v \Rightarrow x_u$
  - $x_v \Rightarrow x_w$
  - $\neg x_v \Rightarrow \neg x_u \vee \neg x_w$

Reduction: CIRCUIT-SAT $\leq_P$ 3-SAT

- Clause $C = x_v$ for input bits v fixed to one
- Clause $C = \neg x_v$ for input bits v fixed to zero
- Clause $C = x_o$ for output bit
- This formula satisfiable iff circuit is satisfiable.
- But it has clauses of size 1 and 2. Convert to 3-SAT formula by introducing two new variables and clauses that force them to be equal to zero.