Lecture T5: NP-Completeness

Can you color each of the 48 states red, white, or blue so that no two adjacent states have the same color?

Overview

Lecture T3:
- What is an algorithm?
  - Turing machine
- Which problems can be solved on a computer?
  - not the halting problem

Lecture T4:
- Which algorithms will be useful in practice?
  - polynomial vs. exponential algorithms

This lecture:
- Which problems can be solved in practice?
  - probably not 3-COLOR or TSP

Some Hard Problems

3-COLOR: Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

YES instance.

3-COLOR: Given a planar map, can it be colored using 3 colors so that no adjacent regions have the same color?

NO instance.
Some Hard Problems

CIRCUIT-SAT: Is there a way to assign inputs to a given Boolean (combinational) circuit that makes it true?

YES instance.  NO instance.

Some Hard Problems

FACTOR: Given two positive integers $x$ and $U$, is there a nontrivial factor of $x$ that is less than $U$?

- Factoring is at the heart of RSA encryption.

Example 1: $x = 23,536,481,273$, $U = 110,000$.
  - YES: $x = 224,737 \times 104,729$.

Example 2: $x = 23,536,481,277$, $U = 110,000$.
  - NO: $x$ is prime.

Some Hard Problems

TSP: A travelling salesperson needs to visit $N$ cities. Is there a route of length at most $D$?

More Hard Problems

More hard computational problems.

- Biology: protein folding.
- Chemistry: chemical synthesis.
- Civil engineering: equilibrium of urban traffic flow.
- Finance: find minimum risk portfolio of given return.
- Electrical engineering: VLSI layout.
- Medicine: reconstructing 3-D shape from biplane angiocardiogram.
- Operations research: optimal resource allocation.
- Physics: anti-ferromagnetic Potts model.
- Politics: Shapley-Shubik voting power.
- Pop culture: Minesweeper consistency.
- Statistics: optimal experimental design.
Properties of Algorithms

A given problem can be solved by many different algorithms (TMs).
- Which ones are useful in practice?

A working definition: (Jack Edmonds, 1962)
- Efficient: polynomial time for ALL inputs.
  - mergesort requires $N \log_2 N$ steps
- Inefficient: "exponential time" for SOME inputs.
  - brute force TSP takes $N! > 2^N$ steps

Robust definition has led to explosion of useful algorithms for wide spectrum of problems.

Exponential Growth

Exponential growth dwarfs technological change.
- Suppose each electron in the universe had power of today's supercomputers.
- And each works for the life of the universe in an effort to solve TSP problem using brute force $N!$ algorithm from Lecture P6.

Some Numbers

<table>
<thead>
<tr>
<th>quantity</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home PC instructions/second</td>
<td>$10^9$</td>
</tr>
<tr>
<td>Supercomputer instructions per second</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>Seconds per year</td>
<td>$10^9$</td>
</tr>
<tr>
<td>Age of universe in years (estimated)</td>
<td>$10^{13}$</td>
</tr>
<tr>
<td>Electrons in universe (estimated)</td>
<td>$10^{79}$</td>
</tr>
</tbody>
</table>

- Will not succeed for 1,000 city TSP!
  $1000! \gg 10^{1000} \gg 10^{79} \times 10^{13} \times 10^9 \times 10^{12}$

Properties of Problems

Which ALGORITHMS will be useful in practice?
- Efficient: polynomial-time for ALL inputs.
  - broad and robust definition
  - covers virtually all algorithms running on actual computers
- Inefficient: "exponential-time" for SOME inputs.

Which PROBLEMS will we be able to solve in practice?
- Those with efficient algorithms.
- How can I tell if I am trying to solve such a problem?
  - 2-COLOR: yes
  - 3-COLOR: probably no
  - 4-COLOR: yes

Theorem (Appel-Haken, 1976). Every planar map is 4 colorable.

P

Definition of P:
- Set of all decision problems solvable in polynomial time on a deterministic Turing machine.

MULTIPLE: Is the integer $y$ a multiple of $x$?
- YES: $(x, y) = (17, 51)$.

RELPRIME: Are the integers $x$ and $y$ relatively prime?
- YES: $(x, y) = (34, 39)$.

Definition important because of Strong Church-Turing thesis.
Strong Church-Turing Thesis

Strong Church-Turing thesis:
- P is the set of all decision problems solvable in polynomial time on REAL computers.

Evidence supporting thesis:
- True for all physical computers: can create deterministic TM that efficiently simulates any existing digital computer.

Possible exception?
- Quantum computers – no conventional gates.

NP

EXP: set of all decision problems solvable in exponential time on a deterministic Turing machine.

NP: does NOT mean "not polynomial."

NP: set of all decision problems with efficient certification algorithm.
- Efficient: polynomial number of steps on deterministic TM.
- Certifier: algorithm to check whether a proposed "solution" is correct.
  - proposed solution is called CERTIFICATE (a hint)
  - technical condition: certificate must be of polynomial-size.

Certifiers and Certificates

COMPOSITE: Given integer s, is s composite?

Observation. s is composite ⇔ there exists an integer 1 < t < s such that s is a multiple of t.
- YES instance: s = 437,669.
  - certificate t = 541 or 809 (a factor)
  - NO instance: s = 437,677.
  - no witness can fool verifier into saying YES
- Conclusion: COMPOSITE ∈ NP.
Certifiers and Certificates

3-COLOR: Given planar map, can it be colored with 3 colors?

Certifier:
1. Check that s and t describe same map.
2. Count number of distinct colors in t.
3. Check all pairs of adjacent states.

Certificate t:

- YES
- NO

NO

Input s:

Certificate t:

s is a YES instance
no conclusion

3-COLOR ∈ NP.

NP

NP: set of decision problems with efficient certification algorithms.

NP: set of all decision problems solvable in polynomial time on a NONDETERMINISTIC Turing machine.

- Equivalent definition.
- Intuition: nondeterministic TM can guess and check all possible solutions in parallel.
- Real computer can simulate nondeterministic TM, but takes exponential time unless you get "lucky."
  - \( P \subseteq NP \subseteq EXP \)

The Main Question

Does \( P = NP? \)\ (Edmonds, 1962)

- Is the original DECISION problem as easy as CERTIFICATION?
- Does nondeterminism help you solve problems faster?

Most important open problem in computer science.

- If yes, staggering practical significance.
- Clay Foundation Millennium $1 million prize.

The Main Question

Does \( P = NP? \)

- Is the original DECISION problem as easy as CERTIFICATION?

If yes, then:

- Efficient algorithms for 3-COLOR, TSP, FACTOR.
- Cryptography is impossible (except for one-time pads) on conventional machines.
- Modern banking system will collapse.
- Harmonial bliss.

If no, then:

- Can’t hope to write efficient algorithm for TSP.
  - see NP-completeness
- But maybe efficient algorithm still exists for factoring?
The Main Question

Does $P = NP$?
- Is the original DECISION problem as easy as CERTIFICATION?

Probably no, since:
- Thousands of researchers have spent four decades in search of polynomial algorithms for many fundamental NP problems without success.
- Consensus opinion: $P \neq NP$.

But maybe yes, since:
- No success in proving $P \neq NP$ either.

NP-Complete

Definition of NP-complete:
- A problem in NP with the property that if it can be solved efficiently, then it can be used as a subroutine to solve any other problem in NP efficiently.
- “Hardest computational problems” in NP.

EXP $\subseteq$ NP $\subseteq$ EXP

If $P \neq NP$, $P$ can be NP-complete.

If $P = NP$, $P = NP$.

NP-Complete Links together a huge and diverse number of fundamental problems:
- TSP, 3-COLOR, CIRCUIT-SAT, thousands more.
- Given an efficient algorithm for 3-COLOR, can efficiently solve TSP, CIRCUIT-SAT, FACTOR, etc.
- Can implement any program in 3-COLOR.

Note: FACTOR not known to be NP-complete.

Notorious complexity class.
- Only exponential algorithms known for these problems.
- Called intractable - unlikely that they can be solved given limited computing resources.

Reduction

Reduction is a general technique for showing that one problem is harder (easier) than another.
- For problems $Y$ and $X$, we can often show: if $Y$ can be solved efficiently, then so can $X$.
- In this case, we say $X$ reduces to $Y$. ($X$ is “easier” than $Y$).

Warmup: PRIMALITY reduces to FACTOR.
- Given an efficient algorithm for FACTOR($x, U$), want to design an efficient algorithm for PRIMALITY($p$).
  - Step 1: Compute FACTOR($p, p$).
  - Step 2: If answer $= YES$, return NO; otherwise return YES.

- Original problem: Is $p = 437,669$ prime?
The "World’s First" NP-Complete Problem

SAT is NP-complete. (Cook-Levin, 1960s)

Idea of proof:
- Given problem \( X \in \text{NP} \), by definition there exists nondeterministic TM \( M \) that solves \( X \) in polynomial time.
- Use Boolean variables to model which symbol occupies cell \( i \) at step \( t \), location of read head at step \( t \), state of finite control at step \( t \), etc.
- Use logic gates to ensure machine makes legal moves, etc.
- SAT instance is satisfiable if and only if TM outputs YES.

Coping With NP-Completeness

Hope that worst case doesn’t occur.
- Complexity theory deals with worst case behavior. The instance(s) you want to solve may be “easy.”
  - TSP where all points are on a line or circle
  - 13,509 US city TSP problem solved

Coping With NP-Completeness

Hope that worst case doesn’t occur.

Change the problem.
- Develop a heuristic, and hope it produces a good solution.
  - TSP assignment
  - Metropolis algorithm, simulating annealing, genetic algorithms
- Design an approximation algorithm: algorithm that is guaranteed to find a high-quality solution in polynomial time.
  - active area of research, but not always possible!
  - Euclidean TSP tour within 1% of optimal

(Cook et. al., 1998)
Coping With NP-Completeness

- Hope that worst case doesn’t occur.
- Change the problem.
- Exploit intractability.
- Keep trying to prove $P = NP$.

Summary

Many fundamental problems are NP-complete.
- TSP, CIRCUIT-SAT, 3-COLOR.

Theory says we probably won’t be able to design efficient algorithms for NP-complete problems.
- You will likely run into these problems in your scientific life.
- If you know about NP-completeness, you can identify them and avoid wasting time.