Diagonalization

Our goal: separate interesting complexity classes

Question: could we somehow use diagonalization to resolve P v NP?

Only general technique we have (e.g. for hierarchy theorems)

Diagonalization relies on following properties of Turing machines:

- A Turing machine can be represented by a string
- A Turing machine can be simulated by another Turing machine without much overhead in time or space.
- Treats machines as blackboxes: internal workings don't matter.

Could we use diagonalization to resolve P v NP?

Here's some evidence that this won't work:

An Oracle Turing machine M^A is a modified TM with the ability to query an oracle for language A.
Has special tape called an oracle tape.
When M^A writes a string on oracle tape, it is informed whether that string is in A in one step.

Observations:

- NP $\subseteq P^{SAT}$
- coNP $\subseteq P^{SAT}$ (because deterministic complexity classes are closed under complementation)
- NP^{SAT} contains languages we believe are not in NP.
  - Example: $\{\Phi \mid \Phi$ is not a minimal boolean formula$\}$

Relativization

An argument "relativizes" if it goes through when you give the machine oracle access.

Essentially, diagonalization is a simulation of one TM by another.
Simulation ensures that simulating machine determines behavior of other machine and then behaves differently.
What if you add an oracle?
Simulation proceeds as before => if we could prove P $\neq$ NP, we could also prove $P^A \neq NP^A$

Diagonalization and Relativization

Theorem:
There is an oracle B whereby $P^B = NP^B$
There is an oracle A whereby $P^A \neq NP^A$

SPACE: The next frontier

Quite different from time: space can be reused.

Space complexity of Turing machine M = space used.
= maximum number of tape cells that M scans as function of input length.

For non-deterministic TM, wherein all branches halt, space complexity defined as maximum number of tape cells scanned on any branch of computation as function of input length.

As usual, use asymptotic notation.

SPACE(f(n)) = {L | L is language decided by an O(f(n)) space deterministic TM}
NSPACE(f(n)) = {L | L is language decided by an O(f(n)) space nondeterministic TM}

Savitch's Theorem

Savitch's Thm: Any nondeterministic TM that uses f(n) space can be converted to deterministic TM that uses $O(f^2(n))$ space.

Idea: Solve yieldability problem.
Given two configurations of the NTM *C* and *C'*, together with number t, determine if NTM can get from *C* to *C'* in t steps.

Solve CanYield(C_{start}, C_{accept}, $2^{O(f(n))}$)

Use recursive algorithm, by searching for intermediate configuration.

Savitch's theorem

```
boolean CanYield(c1, c2, t) {
   if (t ≤ 1) return correct answer
                 enumerate using binary counter
   foreach configuration c' {
      boolean x = CanYield(c1, c', t/2)
      boolean y = CanYield(c2, c', t/2)
      if (x and y) return true
   }
   return false
}
```

On input w:

Output the result of CanYield (c_{start}, c_{accept}, $2^{O(f(n))}$)

- Whenever CanYield invokes itself recursively, stores the current stage number and values of c1, c2 and t on stack.
- Every level of recursion uses O(f(n)) space.
- Depth of recursion O(f(n))
  => total space $O(f(n)^2)$

PSPACE

P. Decision problems solvable in polynomial time.

PSPACE. Decision problems solvable in polynomial space.

EXPTIME. Decision problems solvable in exponential time.

Relationships: P $\subseteq$ NP $\subseteq$ PSPACE = NPSPACE

PSPACE

Binary counter. Count from 0 to 2^n - 1 in binary.
Algorithm. Use n bit odometer.

Claim. 3-SAT is in PSPACE.
Pf.
- Enumerate all 2^n possible truth assignments using counter.
- Check each assignment to see if it satisfies all clauses. ▪

Theorem. NP $\subseteq$ PSPACE.
Pf. Consider arbitrary problem Y in NP.
- Since Y $\leq_P$ 3-SAT, there exists algorithm that solves Y in poly-time plus polynomial number of calls to 3-SAT black box.
- Can implement black box in poly-space. ▪

Quantified Satisfiability

Quantified Satisfiability

QSAT. Let $\Phi(x_1, \ldots, x_n)$ be a Boolean CNF formula. Is the following propositional formula true?

$$\exists x_1 \, \forall x_2 \, \exists x_3 \, \forall x_4 \ldots \, \forall x_{n-1} \, \underset{\substack{\uparrow \\ \text{assume n is odd}}}{\exists x_n} \, \Phi(x_1, \ldots, x_n)$$

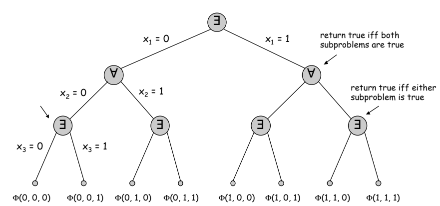
Intuition. Amy picks truth value for x_1, then Bob for x_2, then Amy for x_3, and so on. Can Amy satisfy Φ no matter what Bob does?

Ex. $(x_1 \vee x_2) \wedge (x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$
Yes. Amy sets x_1 true; Bob sets x_2; Amy sets x_3 to be same as x_2.

Ex. $(x_1 \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3)$
No. If Amy sets x_1 false; Bob sets x_2 false; Amy loses;
if Amy sets x_1 true; Bob sets x_2 true; Amy loses.

QSAT is in PSPACE

Theorem. QSAT $\in$ PSPACE.
Pf. Recursively try all possibilities.
- Only need one bit of information from each subproblem.
- Amount of space is proportional to depth of function call stack.



Planning

Planning Problem

Conditions. Set $C = \{ C_1, \ldots, C_n \}$.
Initial configuration. Subset $c_0 \subseteq C$ of conditions initially satisfied.
Goal configuration. Subset $c^* \subseteq C$ of conditions we seek to satisfy.
Operators. Set $O = \{ O_1, \ldots, O_k \}$.

- To invoke operator O_i, must satisfy certain prereq conditions.
- After invoking O_i certain conditions become true, and certain conditions become false.

PLANNING. Is it possible to apply sequence of operators to get from initial configuration to goal configuration?

Examples.

- Many puzzles such as 15-puzzle, Rubik's cube.
- Logistical operations to move people, equipment, materials and robots (software or hardware).

Planning Problem: Binary Counter

Planning example. Can we increment an n-bit counter from the all-zeroes state to the all-ones state?

Conditions. $C_1, \ldots, C_n$. ← C_i corresponds to bit i = 1
Initial state. $c_0 = \phi$. ← all 0s
Goal state. $c^* = \{C_1, \ldots, C_n \}$. ← all 1s
Operators. $O_1, \ldots, O_n$.

- To invoke operator O_i, must satisfy $C_1, \ldots, C_{i-1}$. ← i-1 least significant bits are 1
- After invoking O_i, condition C_i becomes true. ← set bit i to 1
- After invoking O_i, conditions $C_1, \ldots, C_{i-1}$ become false. ← set i-1 least significant bits to 0

Solution. $\{C_1\} \Rightarrow \{C_2\} \Rightarrow \{C_1, C_2\} \Rightarrow \{C_3\} \Rightarrow \{C_3, C_1\} \Rightarrow \ldots$

Observation. Any solution requires at least $2^n - 1$ steps.

Planning Problem: In Exponential Time

Configuration graph G.

- Include node for each of 2^n possible configurations.
- Include an edge from configuration c' to configuration c'' if one of the operators can convert from c' to c''.

PLANNING. Is there a path from c_0 to c^* in configuration graph?

Claim. PLANNING is in EXPTIME.
Pf. Run BFS to find path from c_0 to c^* in configuration graph. ▪

Note. Configuration graph can have 2^n nodes, and shortest path can be of length = $2^n - 1$.

Planning Problem: In Polynomial Space

Theorem. PLANNING is in PSPACE.
Pf. Same idea as proof of Savitch's theorem.

- Suppose there is a path from c_1 to c_2 of length L.
- Path from c_1 to midpoint and from c_2 to midpoint are each ≤ L/2.
- Enumerate all possible midpoints.
- Apply recursively. Depth of recursion = $\log_2$ L. ▪

```
boolean hasPath(c1, c2, L) {
   if (L ≤ 1) return correct answer

                enumerate using binary counter
   foreach configuration c' {
      boolean x = hasPath(c1, c', L/2)
      boolean y = hasPath(c2, c', L/2)
      if (x and y) return true
   }
   return false
}
```


PSPACE-Completeness

PSPACE-Complete

PSPACE. Decision problems solvable in polynomial space.

PSPACE-Complete. Problem Y is PSPACE-complete if (i) Y is in PSPACE and (ii) for every problem X in PSPACE, $X \le_P Y$.

Why polynomial reducibility?

Think about what it means for a problem to be **complete for a complexity class**

- one of the hardest problems in the class
- every other problem in the class *easily* reduced to it.
- so reduction must be easy, relative to complexity of typical problems in class.

PSPACE-Complete

PSPACE. Decision problems solvable in polynomial space.

PSPACE-Complete. Problem Y is PSPACE-complete if (i) Y is in PSPACE and (ii) for every problem X in PSPACE, $X \le_P Y$.

Theorem. [Stockmeyer-Meyer 1973] QSAT is PSPACE-complete.

Theorem. PSPACE $\subseteq$ EXPTIME.
Pf. Previous algorithm solves QSAT in exponential time, and QSAT is PSPACE-complete. ▪

Summary. P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME.

it is known that P ≠ EXPTIME, but unknown which inclusion is strict; conjectured that all are

PSPACE-Complete Problems

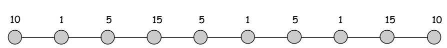
More PSPACE-complete problems.

- Competitive facility location.
- Natural generalizations of games.
  - Othello, Hex, Geography, Rush-Hour, Instant Insanity
  - Shanghai, go-moku, Sokoban
- Various motion planning and search problems.

Competitive Facility Location

Input. Graph with positive edge weights, and target B.
Game. Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors has been selected.

Competitive facility location. Can second player guarantee at least B units of profit?



Yes if B = 20; no if B = 25.

Competitive Facility Location

Claim. COMPETITIVE-FACILITY is PSPACE-complete.

Pf.

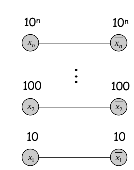
- In PSPACE
- To show that it's complete, we show that QSAT polynomial reduces to it. Given an instance of QSAT, we construct an instance of COMPETITIVE-FACILITY such that player 2 can force a win iff QSAT formula is true.

Competitive Facility Location

assume n is odd

Construction. Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ of QSAT.

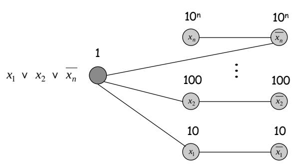
- Include a node for each literal and its negation and connect them.
  - at most one of x_i and its negation can be chosen
- Choose $c \ge k+2$, and put weight c^i on literal x^i and its negation; set $B = c^{n-1} + c^{n-3} + \ldots + c^4 + c^2 + 1$.
  - ensures variables are selected in order $x_n, x_{n-1}, \ldots, x_1$.
- As is, player 2 will lose by 1 unit: $c^{n-1} + c^{n-3} + \ldots + c^4 + c^2$.



Competitive Facility Location

Construction. Given instance $\Phi(x_1, \ldots, x_n) = C_1 \wedge C_1 \wedge \ldots C_k$ of QSAT.

- Give player 2 one last move on which she can try to win.
- For each clause C_j, add node with value 1 and an edge to each of its literals.
- Player 2 can make last move iff truth assignment defined alternately by the players failed to satisfy some clause. ▪



Other important results related to space complexity

Sipser, Sections 8.4-- 8.6; Arora, Barak, Section 3.4

- The classes L (logspace) and NL (nondeterministic logspace)

.

- NL completeness
- NL= coNL

Sipser, Section 9.1; Arora, Barak, Section 4.2

- Space hierarchy theorem
- Corollaries:
  - NL strictly contained in PSPACE
  - PSPACE strictly contained in EXPSPACE