Chapter 9

PSPACE: A Class of Problems Beyond NP
**Geography Game**

**Geography.** Alice names capital city \( c \) of country she is in. Bob names a capital city \( c' \) that starts with the letter on which \( c \) ends. Alice and Bob repeat this game until one player is unable to continue. Does Alice have a forced win?

Ex. Budapest → Tokyo → Ottawa → Ankara → Amsterdam → Moscow → Washington → Nairobi → ...

**Geography on graphs.** Given a directed graph \( G = (V, E) \) and a start node \( s \), two players alternate turns by following, if possible, an edge out of the current node to an unvisited node. Can first player guarantee to make the last legal move?

**Remark.** Some problems (especially involving 2-player games and AI) defy classification according to P, EXPTIME, NP, and NP-complete.
9.1 PSPACE
P. Decision problems solvable in polynomial time.

PSPACE. Decision problems solvable in polynomial space.

Observation. $P \subseteq PSPACE$.

poly-time algorithm can consume only polynomial space
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.
9.3 Quantified Satisfiability
Quantified Satisfiability

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

$$
\exists x_1 \ \forall x_2 \ \exists x_3 \ \forall x_4 \ldots \ \forall x_{n-1} \ \exists x_n \ \Phi(x_1, \ldots, x_n)
$$

\[ \uparrow \]

assume $n$ is odd

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

**Ex.** $(x_1 \lor x_2) \land (x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3)$

**Yes.** Amy sets $x_1$ true; Bob sets $x_2$; Amy sets $x_3$ to be same as $x_2$.

**Ex.** $(x_1 \lor x_2) \land (\overline{x_2} \lor \overline{x_3}) \land (\overline{x_1} \lor \overline{x_2} \lor x_3)$

**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.
**Theorem.** $\text{QSAT} \in \text{PSPACE}$.

**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.

```
x_1 = 0
x_1 = 1
x_2 = 0
x_2 = 1
x_3 = 0
x_3 = 1
Φ(0, 0, 0)  Φ(0, 1, 0)  Φ(1, 0, 0)  Φ(1, 1, 0)
Φ(0, 0, 1)  Φ(0, 1, 1)  Φ(1, 0, 1)  Φ(1, 1, 1)
```

return true iff both subproblems are true

return true iff either subproblem is true

Diagram:

- Decision tree for QSAT
- Nodes represent decision points
- Leaves represent truth assignments
- Edges indicate variable assignments
- Path from root to leaf represents a solution
9.4 Planning Problem
8-puzzle, 15-puzzle. [Sam Loyd 1870s]

- Board: 3-by-3 grid of tiles labeled 1-8.
- Legal move: slide neighboring tile into blank (white) square.
- Find sequence of legal moves to transform initial configuration into goal configuration.
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.**

- 15-puzzle.
- Rubik's cube.
- Logistical operations to move people, equipment, and materials.
Planning Problem: 8-Puzzle

Planning example. Can we solve the 8-puzzle?

Conditions. $C_{ij}, 1 \leq i, j \leq 9$. \( \leftarrow \) $C_{ij}$ means tile \( i \) is in square \( j \)

Initial state. \( c_0 = \{C_{11}, C_{22}, \ldots, C_{66}, C_{78}, C_{87}, C_{99}\} \).

Goal state. \( c^* = \{C_{11}, C_{22}, \ldots, C_{66}, C_{77}, C_{88}, C_{99}\} \).

Operators.
- Precondition to apply $O_i = \{C_{11}, C_{22}, \ldots, C_{66}, C_{78}, C_{87}, C_{99}\}$.
- After invoking $O_i$, conditions $C_{79}$ and $C_{97}$ become true.
- After invoking $O_i$, conditions $C_{78}$ and $C_{99}$ become false.

Solution. No solution to 8-puzzle or 15-puzzle!
Diversion: Why is 8-Puzzle Unsolvable?

8-puzzle invariant. Any legal move preserves the parity of the number of pairs of pieces in reverse order (inversions).

\[ \begin{array}{ccc}
3 & 1 & 2 \\
4 & 5 & 6 \\
8 & 7 \\
\end{array} \quad \rightarrow \quad \begin{array}{ccc}
3 & 1 & 2 \\
4 & 5 & 6 \\
8 & 7 \\
\end{array} \quad \rightarrow \quad \begin{array}{ccc}
3 & 1 & 2 \\
4 & 6 \\
8 & 5 & 7 \\
\end{array} \]

3 inversions 1-3, 2-3, 7-8
3 inversions 1-3, 2-3, 7-8
5 inversions 1-3, 2-3, 7-8, 5-8, 5-6

\[ \begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 \\
\end{array} \quad \rightarrow \quad \begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
8 & 7 \\
\end{array} \]

0 inversions
1 inversion: 7-8
Planning Problem: Binary Counter

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

Conditions. $C_1, \ldots, C_n$. \hspace{1cm} $C_i$ corresponds to bit $i = 1$

Initial state. $c_0 = \phi$. \hspace{1cm} all 0s

Goal state. $c^* = \{C_1, \ldots, C_n\}$. \hspace{1cm} all 1s

Operators. $O_1, \ldots, O_n$.

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

Solution. \{\} $\Rightarrow$ \{C\} $\Rightarrow$ \{C\} $\Rightarrow$ \{C, C\} $\Rightarrow$ \{C\} $\Rightarrow$ \{C\} $\Rightarrow$ ... 

Observation. Any solution requires at least $2^n - 1$ steps.
Planning Problem: In Exponential Space

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 $= 2^n - 1$. 

binary counter
Planning Problem: In Polynomial Space

**Theorem.** PLANNING is in PSPACE.

**Pf.**

- 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 $\leq L/2$.
- Enumerate all possible midpoints.
- Apply recursively. Depth of recursion $= \log_2 L$. □

```java
boolean hasPath(c1, c2, L) {
    if (L ≤ 1) return correct answer
    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
}
```
9.5 PSPACE-Complete
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, $X \leq_p Y$.

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

\[
\uparrow \quad \uparrow \quad \uparrow
\]

it is known that $P \neq EXPTIME$, 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
- Given a memory restricted Turing machine, does it terminate in at most $k$ steps?
- Do two regular expressions describe different languages?
- Is it possible to move and rotate complicated object with attachments through an irregularly shaped corridor?
- Is a deadlock state possible within a system of communicating processors?
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.

- To solve in poly-space, use recursion like QSAT, but at each step there are up to $n$ choices instead of 2.

- 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

Construction. Given instance $\Phi(x_1, \ldots, x_n) = C_1 \land C_1 \land \ldots \land C_k$ 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 $c \geq k+2$, 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: $c^{n-1} + c^{n-3} + \ldots + c^4 + c^2$. 

![Diagram](attachment:image.png)
Competitive Facility Location

**Construction.** Given instance \( \Phi(x_1, \ldots, x_n) = C_1 \land C_1 \land \ldots \land C_k \) 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. □