Detailed Explanation of the Complexity of the 2-SAT Problem

Manav Mantry

Problem Statement

Is the 2-SAT problem NP-Hard? Can it be solved in polynomial time? Explain your reasoning.

Introduction

The Boolean Satisfiability Problem (SAT) is the first known NP-Complete problem. Its complexity depends heavily on the number of literals allowed in each clause.

The **2-SAT problem** is a restricted version of SAT in which each clause contains at most two literals. Despite being a satisfiability problem, 2-SAT turns out to be much easier than general SAT or 3-SAT.

Why 2-SAT Is Not NP-Hard

A problem is NP-Hard if every problem in NP can be reduced to it in polynomial time. General SAT and 3-SAT satisfy this property.

However, restricting clauses to only two literals significantly limits the logical relationships that can be expressed. As a result:

- 2-SAT lacks the expressive power needed to encode arbitrary NP problems
- Many reductions used to prove NP-Hardness fail when clauses are limited to two literals

If 2-SAT were NP-Hard, then it would imply that all NP problems could be solved in polynomial time, which would mean:

$$\text{P} = \text{NP}$$

Since this is widely believed to be false, 2-SAT is not NP-Hard.

Polynomial-Time Solvability of 2-SAT

Unlike 3-SAT, the 2-SAT problem has a well-known polynomial-time algorithm based on graph theory.

Implication Graph Construction

Each clause of the form:

$$(x \vee y)$$

can be rewritten using logical equivalence as:

$$(\neg x \Rightarrow y) \wedge (\neg y \Rightarrow x)$$

Using this idea, we construct a directed graph called the **implication graph**:

- Each variable x gives rise to two vertices: x and $\neg x$
- Each clause adds two directed edges corresponding to its implications

Strongly Connected Components (SCCs)

A key observation is that:

- If both x and $\neg x$ lie in the same strongly connected component, then the formula is unsatisfiable
- Otherwise, a satisfying assignment exists

This condition arises because mutual reachability would force a variable to be both true and false simultaneously.

Algorithmic Implications

Strongly connected components can be found using algorithms such as:

- Kosaraju's algorithm
- Tarjan's algorithm

Both algorithms run in linear time with respect to the size of the graph.

Time Complexity

Let n be the number of variables and m the number of clauses.

The implication graph has:

- $2n$ vertices
- $2m$ edges

Thus, the total time complexity is:

$$O(n + m)$$

which is polynomial (in fact, linear).

Key Insight: 2-SAT vs 3-SAT

The dramatic difference in complexity between 2-SAT and 3-SAT illustrates how adding just one extra literal per clause increases computational difficulty:

$$\text{2-SAT} \in \text{P}, \quad \text{3-SAT} \in \text{NP-Complete}$$

Conclusion

2-SAT is not NP-Hard and can be solved efficiently in polynomial time.

This result is a fundamental example in computational complexity showing how small changes in problem constraints can lead to major differences in solvability.