Abstract

The P vs NP problem remains one of the most profound open questions in computer science, with far-reaching implications across computational theory, cryptography, and complexity analysis. In this study, we introduce a novel approach to the P vs NP problem by analyzing the structure of *unverified solution sets* generated during the process of solving decision problems. By classifying these sets based on their cardinality—finite, countably infinite, or uncountable—and analyzing the growth rate of candidate solutions relative to input size, we offer a new framework for interpreting the boundary between tractable and intractable problems. Our results suggest that the computational feasibility of a problem can be directly linked to the properties of its unverified solution set, offering deeper insight into the theoretical underpinnings of P vs NP.

Introduction

The P vs NP problem is one of the seven Millennium Prize Problems announced by the Clay Mathematics Institute. Despite decades of research, it remains unresolved. At its core, the question asks whether every problem whose solution can be verified quickly (in polynomial time) can also be solved quickly.

This report proposes a structural framework to address the P vs NP question by examining the nature of unverified solution sets that arise from solving NP problems. The hypothesis is that the relationship between P and NP depends on whether the structure and size of the candidate solution space permit efficient verification and discovery of valid answers.

P &NP

I Definitions and Background

I.1 Class P:

P (Polynomial time) consists of decision problems for which a solution can be found in polynomial time using a deterministic Turing machine. These are considered efficiently solvable problems.

I.2 Class NP:

NP (Nondeterministic Polynomial time) consists of decision problems for which a proposed solution can be verified in polynomial time, although finding such a solution may not be efficient.

I.3 Unverified Solution Sets:

Given a decision problem A and an algorithm B designed to solve it, let C = $\{c_0; c_1; ...; c_i\}$ be the set of candidate answers generated by B before verification. We define this as the ***unverified solution set***.

I.4 The Set of Unverified Candidate Solutions in NP:

Consider a problem A∈NP and B is an algorithm designed to solve A. We define C as the set of unverified candidate solutions generated through algorithm B for problem A.

II Relationship Between A and P.

To address the P vs NP problem, we need to observe the relationship between A and P. Thus, we assume B is the best algorithm among all algorithms designed to solve A (we are not concerned here with how to find B). We will analyze C in three cases as follows:

- **C is a finite set.**

  In this case, we can verify the correct solutions in C. Therefore, A can be considered a problem that can be solved quickly..

At this point, we can determine whether A$\in P$, or we can assume P = NP

Consider the following example to clarify this:

Example: Given x $\in N$, x $\leq$ 1000, find x such that $x^2 = 625$.

Algorithm:: $\sqrt{625} = 25$ so x= 25 or x= -25

Since x $\in N$, we have x = 25.

From this example, we observe that even if the input value increases indefinitely or x$\in R$. the value of C = {-25,25}. In other words C is finite.

- C is an uncountable set.

If C is uncountable, such as the set of real numbers, then it is impossible to enumerate and test all candidate solutions. Therefore, the problem is in NP (verifiable) but not in P (not efficiently solvable), implying P ≠ NP.

Example: The Real Number Safe Password Problem

Harry is a billionaire and the chairman of a major bank. At home, he has a safe protected by the most advanced security technology available. The safe can only be opened by entering the correct password—a real number.

Andy, a notorious thief with extraordinary talent, learns from the maid that the password is a real number, but he obtains no additional information. One day, while Harry is away on a business trip, Andy sneaks into the room where the safe is kept and faces the challenge of unlocking it.

The problem is as follows: Can Andy find the correct password—that is, a specific real number—to open the safe?

Verifying whether a guessed value is the correct password is very quick (he simply enters the number and waits to see if the safe opens).

However, finding the correct password is impossible without further information, since the set of real numbers is uncountable.

- Countably Infinite Unverified Set.

Here, we analyze the density of C relative to input size n:

Let:

- n be the input size
- C_n be the number of unverified candidates for input size n
- d = $\frac{C_n}{n}$

If d ≤ 1, the problem may still be in P, suggesting P = NP.

If d > 1, we consider two subcases:

1. **d is a constant** When d is a constant greater than 1, the feasibility of solving A depends on computational power. Define a threshold a; if d ≤ a, A ∈ P. Otherwise, A ∈ NP \ P.
2. **d grows exponentially (e.g., d = b^n for b > 1)** As input size increases, the number of candidates grows exponentially, making exhaustive checking infeasible. Hence, A ∈ NP \ P, and P ≠ NP.

III Summary of Conditions. We summarize the relationship between P and NP based on unverified solution set C as follows:

- P = NP when:
  - C is finite
  - C is countably infinite with $d \leq 1$
- P ≠ NP when:
  - C is uncountable
  - C is countably infinite with $d = b^n$, $b > 1$
- In ambiguous cases where $d = a$ (constant), the classification depends on the growth of computational power.

Conclusion

This research presents a new perspective on the P vs NP problem by focusing on the nature of unverified solution sets associated with decision problems in NP. Through a structural classification of these sets and the introduction of a density metric $d = \frac{C_n}{n}$, we propose a means to evaluate whether a given NP problem admits a polynomial-time solution. The analysis shows that problems with finite or sufficiently sparse countably infinite candidate spaces may be tractable and belong to class P. Conversely, problems with uncountable or rapidly expanding solution spaces (e.g., exponential growth in d) are likely to remain intractable. These insights not only reinforce the distinction between P and NP under certain conditions but also suggest that the resolution of P vs NP may depend on a deeper understanding of solution space complexity. Future work may further formalize the relationship between d, algorithmic verification, and computational feasibility.

Bibliography