# A Survey on P versus NP problem

A.Z

This article delves into the enigmatic $\mathcal{P}$ *vs*. $\mathcal{NP}$ problem, a cornerstone of theoretical computer science, and helps readers get a glimpse of the definition of $\mathcal{P}$ and $\mathcal{NP}$ class and what the problem means. This paper begins by tracing the origins of the problem, then shifts to the fundamentals of computational complexity theory, and setting the stage for a deeper understanding of the $\mathcal{P}$ and $\mathcal{NP}$ classes. The paper explores the intriguing relationship between these classes, pondering the profound implications of whether $\mathcal{P}$ equals $\mathcal{NP}$ or not. The narrative further extends to the realms of $\mathcal{NP}$-complete ($\mathcal{NPC}$), $\mathcal{NP}$-hard, and beyond, including exponential time ($\mathcal{EXP}$) and undecidable problems, thereby painting a comprehensive picture of the complexity landscape. This survey also touches upon the historical and practical significance of these problems, their impact on fields like cryptography, and the philosophical questions they raise about the nature of problem-solving and computational limits.

Additional Key Words and Phrases: P, NP, NP-Completeness, NP-Hard, Computational Theory

**ACM Reference Format:**
A.Z. 2023. A Survey on P versus NP problem. *ACM Trans. Graph.* 00, 0, Article 000 ( 2023), 4 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

The $\mathcal{P}$ *vs*. $\mathcal{NP}$ problem is a renowned problem in theoretical computer sciences. It is also one of the Millennium Prize Problems selected by the Clay Mathematics Institute in 2000, which means proofing (or disproofing) it can not only gain reputation but also one million dollars. This problem appeared explicitly first in the paper of [Cook 1971] and [Левин 1973]. In a nutshell, the essence of the $\mathcal{P}$ *vs*. $\mathcal{NP}$ problem is to determine whether every problem whose solution can be quickly verified (denoted $\mathcal{NP}$) also has a solution that can be efficiently discovered (denoted $\mathcal{P}$ ). For example, it is easy to check whether a password is correct or not, and intuitively finding the correct password is harder. The $\mathcal{P}$ *vs*. $\mathcal{NP}$ problem seeks to determine whether the process of finding a solution, such as a password, can be as efficient as the process of verifying that password.

The present paper aims to demystify the $\mathcal{P}$ vs. $\mathcal{NP}$ problem, a fundamental problem in theoretical computer science, by first offering a background of its origins and the essential concepts of computational complexity theory. Then, the paper moves onto a detailed examination of $\mathcal{NP}$-complete ($\mathcal{NPC}$), $\mathcal{NP}$-hard, and other related categories like exponential time ($\mathcal{EXP}$) and undecidable problems, thereby providing a holistic view of the complexity landscape. Finally, insights into the historical significance, practical applications in fields such as cryptography, and the philosophical questions posed by these problems.

Author's address: A.Z.


0730-0301/2023/0-ART000 $15.00
https://doi.org/XXXXXXX.XXXXXXX

Table 1. Table of common time complexities

| Name | Running time(T(n)) | Example algorithms |
|---|---|---|
| constant time | $O(1)$ | Calculating $1^n$ |
| logarithmic time | $O(\log n)$ | Binary search |
| linear time | $O(n)$ | Linear search |
| linearithmic time | $O(n \log n)$ | Fast Fourier transform |
| quadratic time | $O(n^2)$ | Bubble sort |
| exponential time | $O(2^n)$ | N-queens problem via brute-force |
| factorial time | $O(n!)$ | TSP via brute-force |

## 2 PRELUDE: COMPUTATIONAL THEORY

In 1936, Alan Turing described an abstract computational device to explore the capabilities and boundaries of computability. These devices, which known as "Turing Machine", consist the foundational models of modern computer. In 1976, after many false proofs and counterexamples, a research [Appel and Haken 1976] proved "four color theorem". It was the first major theorem to be proved using a computer. Under the "provision" of Moore's Law, the power of computation has seen a significant rise, while the cost of computing has substantially fallen, not to mention the capabilities brought about by the Internet. Computer is now a commonly used tool in just about every acadecmic field. One of the key factors scientists consider when using computers to solve problems is time complexity.

Time complexity is the computational complexity that describes the amount of computer time it takes to run an algorithm[Wikipedia contributors 2023b]. It does not mean how much time it takes for a algorithm to solve the problem, but how fast the length of time required for the program to grow when the scale of the algorithm expands. The time complexity is commonly expressed using big O notation. For example, $O(1)$ time complexity refers to it always takes so much time for program processing no matter how big the data is, which is good. $O(n)$ time complexity means the time required for calculation is linearly related to the scale of data, algorithms such as finding the maximum value in $n$ numbers is $O(n)$ time complexity. Some algorithms, like bubble sorting, insertion sorting, etc., belong to the complexity of $O(n^2)$, which is more time-consuming than $O(n)$. Also some exhaustive algorithms, the length of the time required increases to the geometric order, which is the exponential complexity of $O(a^n)$, or even $O(n!)$

Table 1 presents a summary of typical time complexities encountered in algorithmic design. In mathematics, $O(1) < O(logn) < O(n) < O(nlogn) < O(n^a) < O(2^n) < O(n!)$.

Among them, complexity less or equal than $O(n^a)$ is called polynomial complexity (or polynomial time). Bigger than $O(n^a)$ is called non-polynomial complexity (or non-polynomial time).

## 3 WHAT DO $\mathcal{P}$ AND $\mathcal{NP}$ MEANS?

The term "$\mathcal{P}$" refers to "polynomial time." In a more formal context, problems classified under the $\mathcal{P}$ class are generally considered to be those that can be solved relatively easily, i.e., a problem belong to $\mathcal{P}$ class if it can be solved within polynomial time.

In paper [Wigderson 2006a], $\mathcal{P}$ class is defined as

**Definition 1:** (The class $\mathcal{P}$ ) A function $f : \mathrm{I} \rightarrow \mathrm{I}$ is in the class $\mathcal{P}$ if there is an algorithm computing $f$ and positive constants $A, c$, such that for every $n$ and every $x \in \mathrm{I}_n$ the algorithm computes $f(x)$ in at most $An^c$ steps.

Algorithm 1 persents Euclid's GCD algorithm, one of the most ancient and famous $\mathcal{P}$ problem.

**Algorithm 1** Calculate GCD of two numbers

**Require:** $A, B$ {Two non-negative integers}
**Ensure:** $GCD(A, B)$
  **if** $A < B$ **then**
    return $GCD(B, A)$
  **end if**
  **if** $A = 0$ **then**
    return $B$
  **end if**
  return $GCD(B, A \bmod B)$

Correspondingly, "$\mathcal{NP}$ " denotes "non-deterministic polynomial time." This classification encompasses a set of problems for which the ease of solution is uncertain, yet the verification of solutions is straightforward. Essentially, the $\mathcal{NP}$ class comprises problems where the verification of a given solution can be executed in polynomial time. However, the time required to actually solve these problems may or may not be polynomial, or it might remain undetermined.

In paper [Wigderson 2006a], $\mathcal{NP}$ class is defined as

**Definition 2:** (The class $\mathcal{NP}$ ) The set $C$ is in the class $\mathcal{NP}$ if there is a function $V_C \in \mathcal{P}$ and a constant $k$ such that

- If $x \in C$ then $\exists y$ with $|y| \leq |x|^k$ and $V_C(x, y) = 1$.
- If $x \notin C$ then $\forall y$ we have $V_C(x, y) = 0$.

Hamiltonian path problem, described as "find a path that visits each vertex of the graph exactly once", is a $\mathcal{NP}$ problem (also a $\mathcal{NP} - complete$ problem). Because it is easy to verify whether a route passes through every vertex. But solving the problem seems to require systematically trying out lots of candidates, which will take non-polynomial time.

## 4 THE $\mathcal{P}$ $vs.$ $\mathcal{NP}$ QUESTION

It is evident that problems in P are also in NP, the verifier of P can be the same as its solver.

$$\mathcal{P} \subseteq \mathcal{NP}$$

Fig1 presents the relationship between $\mathcal{P}$ and $\mathcal{NP}$.

For $\mathcal{P}$ class, which can be solved in polynomial time, we can design a reasonable fast program using computer even the data in this questions are extremely large. All we have to do is wait for computers get more powerful and then, even huge versions of those problems will be easy to solve by a computer. But for some scenarios in $\mathcal{NP}$ class, we can only find a solution with exponential complexity. We do not consider it as a pratical solution because as the data size increases, even if slightly, the time required to obtain this solution is not acceptable, perhaps even forever.

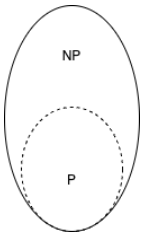


Fig. 1. relationship between P and NP

The $\mathcal{NP}$ class is extremely rich. Thousands of $\mathcal{NP}$ problems exist across various fields such as mathematics, optimization, artificial intelligence, and biology. Efficient algorithms to these problems would offer numerous benefits. Despite decades of attempts, efficient solutions have only succeed for few. Could it be that all problems in the NP class have efficient algorithms that are yet to be discovered? Or maybe we should stop searching because there is no effective method. This the basis of the $\mathcal{P}$ $vs.$ $\mathcal{NP}$ question.

### 4.1 $\mathcal{P} = \mathcal{NP}$ ?

In a popular TV show "The Simpons" episodes "Treehouse of Horror VI", Homer hides behind a bookshelf to avoid his sister-in-law. As it turns out, Homer leaped in an inter-dimensional limbo, one of the mathematic easter eggs show "$\mathcal{P} = \mathcal{NP}$" . The limbo is a completely different place from the world of Homer's itself , which may imply that the author does not think "$\mathcal{P} = \mathcal{NP}$" exists in a normal world.

If all $\mathcal{NP}$ problems are foundational in $\mathcal{P}$, numerous significant puzzles that have posed challenges would become readily solvable by computers. All optimization problems, networking problems, Protein folding problems etc. can quickly get the optimal solution, and even the solution of all mathematical problems can be quickly completed in polynomial time. Additionally, the encryption methods employed for activities such as online banking, which are predicated on $\mathcal{NP}$ problems (primarily prime factorization), would become susceptible to being easily deciphered. But this does not result that life will become easier. According to a survey [Fortnow 2009], even $\mathcal{P} = \mathcal{NP}$ is proved, it wouldn't necessarily suggest that we can get efficient polynomial-time algorithms for all $\mathcal{NP}$ problems.

### 4.2 $\mathcal{P} \neq \mathcal{NP}$ ?

According to an option poll [Gasarch 2019] on "$\mathcal{P}$ $vs.$ $\mathcal{NP}$", a majority of authorities in this field conjectures that there are optimization problems (with easily checkable solutions) that cannot be solved by efficient algorithms i.e. $\mathcal{P} \neq \mathcal{NP}$ ?. But so far people have not come up with any strong basis for this view.

$\mathcal{P} \neq \mathcal{NP}$ is a somewhat idealistic idea. It is the notion that tasks require a degree of creativity beyond the capabilities of a basic computer program. "We admire Wiles' proof of Fermat's

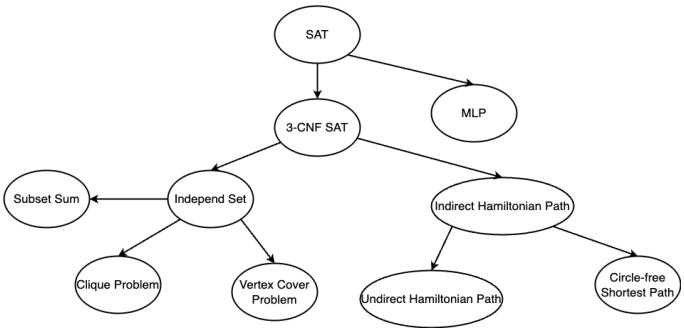


Fig. 2. Some NPC problems in NPC tree

Last Theorem, the scientific theories of Newton, Einstein, Darwin, Watson and Crick, the design of the Golden Gate bridge and the Pyramids, and sometimes even Hercule Poirot's and Miss Marple's analysis of a murder, precisely because they seem to require a leap which cannot be made by everyone, let alone a by simple mechanical device"[Wigderson 2006b].

The challenge in definitively resolving the $\mathcal{P}$ *vs*. $\mathcal{NP}$ problem lies in its inherent complexity. Interestingly, the act of proof $\mathcal{P}$ *vs*. $\mathcal{NP}$ itself constitutes an $\mathcal{NP}$ problem. This presents a paradoxical situation: the difficulty of the problem may be intrinsic, or perhaps it is not yet fully understood. The uncertainty surrounding this issue remains a significant aspect of its study.

## 5 $\mathcal{NPC}$, $\mathcal{NP}-Hard$ AND BEYOND

Afterwards, people discovered a special type of $\mathcal{NP}$ class, which seems to have pushed forward the $\mathcal{P}$ *vs*. $\mathcal{NP}$ problem a little bit. This very special class called $\mathcal{NP}-Complete$, also known as $\mathcal{NPC}$. The $\mathcal{NPC}$ problem is "a slightly more complex" $\mathcal{NP}$ problem, which can be polynomial-time reduced from other $\mathcal{NP}$ problems, i.e., $\mathcal{NPC}$ problems are at least as complex as $\mathcal{NP}$ problems.

In 1971, a pioneering paper [Cook 1971] proved the first $\mathcal{NP}$-complete problem, that is, the satisfiability problem(SAT). Utilizing the principle of transitivity in reductions, to establish that a different problem (for instance, $\mathcal{D} \in \mathcal{NP}$) is $\mathcal{NP}$-Complete, it suffices to construct a reduction from SAT to $\mathcal{D}$, effectively demonstrating that SAT is reducible to $\mathcal{D}$ (denoted as $SAT \leq \mathcal{D}$).

Consequently, following Cook's initial discovery, a steadily increasing number of $\mathcal{NP}$-complete problems have been identified. So far, thousands of $\mathcal{NPC}$ problems have been cataloged. These problems form a massive $\mathcal{NP}$-complete tree, with the root node being Cook's satisfiability problem, shown in 2.

$$SAT \leq D \Rightarrow D \; is \; NPC$$

Now $\mathcal{P}$ *vs*. $\mathcal{NP}$ problems can be transformed into SAT problems. This transformation refers to that if we can prove that any $\mathcal{NPC}$ problem is a $\mathcal{P}$ problem, then we can prove $\mathcal{P} = \mathcal{NP}$.

$$\mathcal{P} = \mathcal{NP} \iff SAT \in \mathcal{P}$$

$\mathcal{NP}$-hard is a class of problems that are informally "at least as hard as the hardest problems in $\mathcal{NP}$" [Wikipedia contributors 2023a]. All $\mathcal{NP}$-complete problems are also $\mathcal{NP}$-hard.

The complexity relationship among the classes $\mathcal{P}$, $\mathcal{NP}$ $\mathcal{NPC}$ and $\mathcal{NP}$-Hard is depicted in Figure 3. It is important to note the position of the boundary points still open.

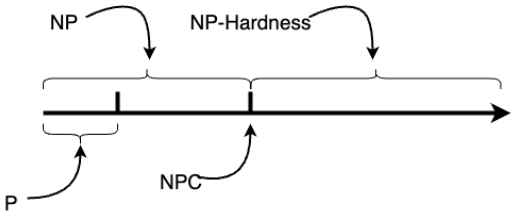


Fig. 3. relationship between P NP NPC and NP-Hard

We can naturally think of a class of problems that cannot be verified in polynomial time and cannot be solved in polynomial time. Take, for instance, the game of chess. Identifying the optimal move in a chess game presents a significant challenge. Even if we were proposed a solution, how can we ascertain the correctness? This class of questions called $\mathcal{EXP}$ (Exponential Time) which includes problems that require exponential time to both solve and verify. Beyond that is the class of undecidable problems, where no algorithm can determine a solution in a finite amount of time, making both computation and verification effectively impossible, e.g. Halting problem. The complexity relationship among the classes is depicted in Figure 4.

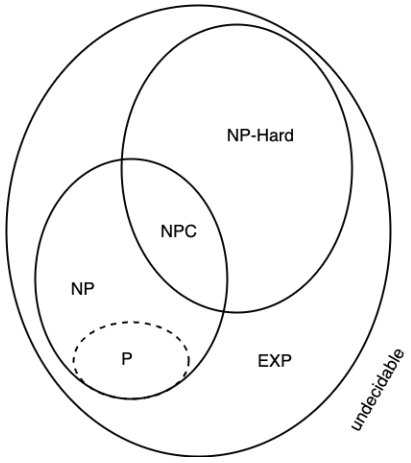


Fig. 4. P NP NPC NP-Hard EXP and undecidable

But sometimes we need to solve the problem anyway. Go is a typical $\mathcal{EXP}$ problem where human players rely on intelligence and intuition. However in 2016, AlphaGo defeated all human Go players. In this scenario, the computer uses a method called Monte Carlo, which does not need to find the best or correct answer, but only needs to find a relatively better answer, better than the human.

## 6 CONCLUSION

The present paper has reviewed the famous $\mathcal{P}$ *vs*. $\mathcal{NP}$ problem, which originated from computer science and expanded to a mathematical problem. This problem is not just looking for what can be

computed in a given amount of space and time. Rather, it delves into the fundamental essence of space and time themselves. This intricate web of computational complexity holds profound implications for fields such as physics and biology, and fundamentally enhances our comprehension of the universe.

## 7 ACKNOWLEDGMENTS

My thanks to 3Blue1Brown and Veritasium, these two YouTube channels have sparked my interest in science and the unknown field.

## REFERENCES

Kenneth Appel and Wolfgang Haken. 1976. The existence of unavoidable sets of geographically good configurations. *Illinois Journal of Mathematics* 20 (1976), 218–297. https://api.semanticscholar.org/CorpusID:117203760

Stephen A. Cook. 1971. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (Shaker Heights, Ohio, USA) *(STOC '71)*. Association for Computing Machinery, New York, NY, USA, 151–158. https://doi.org/10.1145/800157.805047

Lance Fortnow. 2009. The Status of the P versus NP problem. *Commun. ACM* 52 (09 2009), 78–86. https://doi.org/10.1145/1562164.1562186

William Gasarch. 2019. Guest Column: The Third P=?NP Poll. *ACM SIGACT News* 50 (03 2019), 38–59. https://doi.org/10.1145/3319627.3319636

Avi Wigderson. 2006a. P, NP and mathematics - A computational complexity perspective. *Proceedings oh the International Congress of Mathematicians, Vol. 1, 2006-01-01, ISBN 978-3-03719-022-7, pags. 665-712* 1 (01 2006).

Avi Wigderson. 2006b. P, NP and mathematics - A computational complexity perspective. *Proceedings oh the International Congress of Mathematicians, Vol. 1, 2006-01-01, ISBN 978-3-03719-022-7, pags. 665-712* 1 (01 2006), 11.

Wikipedia contributors. 2023a. NP-hardness — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=NP-hardness&oldid=1184694677 [Online; accessed 17-December-2023].

Wikipedia contributors. 2023b. Time complexity — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Time_complexity&oldid=1188334657 [Online; accessed 17-December-2023].

Л. А. Левин. 1973. Универсальные задачи перебора. *Пробл. передачи информ.* 9, 3 (1973), 115–116. https://www.mathnet.ru/links/bafa193f46054869b9c2b7a1dfa5cfe9/ppi914.pdf English translation: Problems of Information Transmission, 9:3 (1973), 265-266.

Received 29 February 2023; revised 32 May 4202; accepted -1 Jabruary 乙巳年