IPI Letters

ISSN 2976 - 730X
IPI Letters 2024,Vol 2 (2):76-79
https://doi.org/10.59973/ipil.122

Received: 2024-09-13
Accepted: 2024-09-14
Published: 2024-09-16

Opinions

# Solving NP-complete Problems Efficiently

**Frank Vega**[1,*]

[1]Independent Researcher, Cotorro, Havana, Cuba

[*]Corresponding author: vega.frank@gmail.com

**Abstract** - The P versus NP problem is a fundamental question in computer science. It asks whether problems whose solutions can be quickly verified can also be quickly solved. Here, "quickly" refers to computational time that grows proportionally to the size of the input (polynomial time). While the problem's roots trace back to a 1955 letter from John Nash, its formalization is attributed to Stephen Cook and Leonid Levin. Despite extensive research, a definitive answer remains elusive. Closely tied to this is the concept of NP-completeness. If a single NP-complete problem could be solved efficiently, it would imply that all problems in NP can be solved efficiently, proving that P equals NP. Garey and Johnson defined K-CLOSURE such that for any edge (u, v) in the directed graph, either node u is in the set V' or node v is not in V'. This implies that either both nodes are in V' or both are not in V'. Our previous work in IPI Letters presented a polynomial-time algorithm for K-CLOSURE. While no errors have been identified in this work, many believe that Garey and Johnson's original definition was incorrect, and their citation of Queyranne was a misunderstanding. Many argue that the empty set serves as a simple counterexample to Garey and Johnson's definition of K-CLOSURE. This paper proposes that K-CLOSURE is actually an NP-complete problem, which would imply that P equals NP.

**Keywords** - Complexity classes; Boolean formula; Graph; Completeness; Polynomial time.

## 1 Introduction

Computer science is confronted by the formidable challenge of the *P* versus *NP* problem [1]. Fundamentally, this inquiry seeks to determine if the ability to swiftly verify a solution implies the capacity to swiftly compute it. Here, "swiftly" denotes algorithms with a polynomial time complexity, where computational time grows proportionally to input size. Problems solvable within polynomial time constitute the class *P*. Conversely, *NP* encompasses problems whose solutions can be verified efficiently given a suitable "certificate" - a piece of information enabling rapid validation [2].

The crux of the *P* versus *NP* question lies in whether *P* and *NP* are identical. A prevailing belief is that *P* is a strict subset of *NP* ($P \neq NP$), signifying that certain problems are inherently more difficult to solve than to verify. Resolving this enigma holds profound implications for fields such as cryptography and artificial intelligence [3], [4]. The *P* versus *NP* problem is widely considered one of the most challenging open questions in computer science. Evidence supporting its difficulty arises from techniques like relativization and natural proofs, which have yielded inconclusive results [5], [6]. Similar problems, such as the VP versus VNP problem in algebraic complexity, remain unsolved [7].

Resolving the *P* versus *NP* question is often described as a "holy grail" of computer science. A positive resolution would revolutionize our understanding of computation, potentially leading to groundbreaking algorithms for critical problems. Reflecting its significance, the problem is listed among the Millennium Prize Problems. While recent years have seen progress in related areas, such

as finding efficient solutions to specific instances of *NP*-complete problems, the core question of *P* versus *NP* remains unanswered [8]. A polynomial-time algorithm for any *NP*-complete problem would directly imply *P* equals *NP* [9]. Our work focuses on presenting such an algorithm for a well-known *NP*-complete problem.

## 2 Background and ancillary results

*NP*-complete problems are the Everest of computational challenges. Despite the ease of verifying proposed solutions with a succinct certificate [9], finding these solutions efficiently remains an elusive goal. A problem is classified as *NP*-complete if it satisfies two stringent criteria within computational complexity theory:

1. **Efficient Verifiability:** Solutions can be swiftly checked using a concise proof.
2. **Universal Hardness:** Every problem in the class *NP* can be transformed into an instance of this problem without significant computational overhead [9].

The implications of finding an efficient algorithm for a single *NP*-complete problem are profound. Such a breakthrough would serve as a master key, unlocking efficient solutions for all problems in *NP*, with transformative consequences for fields like cryptography, artificial intelligence, and planning [3], [4].

Illustrative examples of *NP*-complete problems include:

- **Boolean satisfiability (SAT)**: Given a logical expression, determine if there exists an assignment of truth values to its variables that makes the entire expression true [10].
- **CLIQUE**: Given an undirected graph $G = (V, E)$ and a positive integer $k$, decide whether there exists a subset $V'$ of $V$ containing at least $k$ vertices such that every two vertices in $V'$ are connected by an edge in $G$ [10].

The provided examples represent a small subset of the extensively studied *NP*-complete problems relevant to our current work. We introduce the following problem:

**Definition 2.1.** ***K-CLOSURE***

*INSTANCE: A directed graph $H = (W, A)$ and a positive integer $k$.*

*QUESTION: Is there set $V'$ of at most $k$ vertices such that for all $(u, v) \in A$ either $u \in V'$ or $v \notin V'$?*

*REMARKS: Note that in this problem, the phrase "either $u \in V'$ or $v \notin V'$" is equivalent to either "($u \in V'$ and $v \in V'$) or ($u \notin V'$ and $v \notin V'$)". This is because the logical implication of the phrase "**either ... or ...**" requires that exactly one of the two conditions be true. Vega's paper [11] presents a polynomial-time algorithm for solving this problem. Garey and Johnson asserted in their book [10] that the K-CLOSURE problem is NP-complete.*

By presenting the *NP*-completeness of *K-CLOSURE*, we would establish a proof that *P* equals *NP*.

## 3 Main Result

This is a main insight.

**Theorem 3.1.** *K-CLOSURE ∈ NP–complete.*

*Proof.* The *CLIQUE* problem, which involves determining if a clique of size at least $k$ exists in a given undirected graph $G = (V, E)$, can be reduced to the problem of finding a closure set $V'$ in a constructed directed graph $H = (W, A)$ of size at most $(n - k) + 2 \cdot n^2 \cdot \left(\binom{n}{2} - m + n - k\right)$, where $n =\mid V \mid$ is the number of vertices and $m =\mid E \mid$ is the number of edges of $G$. The directed graph $H$ is created by introducing $2 \cdot n^2$ new vertices $\underbrace{u_{(ab,1)}, u_{(ab,2)}, \ldots, u_{(ab,n^2)}}_{\text{new } n^2 \text{ vertices}} \in W$ and $\underbrace{v_{(ab,1)}, v_{(ab,2)}, \ldots, v_{(ab,n^2)}}_{\text{new } n^2 \text{ vertices}} \in W$ for

each pair of vertices $a, b \in V$ and adding $2 \cdot n^2$ new edges $\underbrace{(a, u_{(ab,1)}), (a, u_{(ab,2)}), \ldots, (a, u_{(ab,n^2)})}_{new\ n^2\ edges} \in A$ and $\underbrace{(b, v_{(ab,1)}), (b, v_{(ab,2)}), \ldots, (b, v_{(ab,n^2)})}_{new\ n^2\ edges} \in A$ whenever $(a, b) \notin E$ or $a = b$. Note that, an empty closure can imply the maximum clique in a complete graph $G$. Given that *CLIQUE* is an *NP*-complete problem, it follows that *K-CLOSURE* is also *NP*-complete. □

This is the main theorem.

**Theorem 3.2.** *$P = NP$.*

*Proof.* A polynomial-time solution to any *NP*-complete problem would establish the equivalence of *P* and *NP* [9]. Despite extensive research on over 300 significant *NP*-complete problems, no such polynomial-time algorithm has been discovered [9]. Given that *K-CLOSURE* is an *NP*-complete problem, a polynomial-time solution for it, as presented in the reference [11], would directly imply *P* equals *NP*. □

## 4 Conclusion

A definitive proof that *P* equals *NP* would fundamentally reshape our computational landscape. The implications of such a discovery are profound and far-reaching:

- **Algorithmic Revolution.**
  - The most immediate impact would be a dramatic acceleration of problem-solving capabilities. Complex challenges currently deemed intractable, such as protein folding, logistics optimization, and certain cryptographic problems, could become efficiently solvable [3]. This breakthrough would revolutionize fields from medicine to cybersecurity. Moreover, everyday optimization tasks, from scheduling to financial modeling, would benefit from exponentially faster algorithms, leading to improved efficiency and decision-making across industries [3].
- **Scientific Advancements.**
  - Scientific research would undergo a paradigm shift. Complex simulations in fields like physics, chemistry, and biology could be executed at unprecedented speeds, accelerating discoveries in materials science, drug development, and climate modeling [3]. The ability to efficiently analyze massive datasets would provide unparalleled insights in social sciences, economics, and healthcare, unlocking hidden patterns and correlations [3].
- **Technological Transformation.**
  - Artificial intelligence would be profoundly impacted. The development of more powerful AI algorithms would be significantly accelerated, leading to breakthroughs in machine learning, natural language processing, and robotics [8]. While the cryptographic landscape would face challenges, it would also present opportunities to develop new, provably secure encryption methods [8].
- **Economic and Societal Benefits.**
  - The broader economic and societal implications are equally significant. A surge in innovation across various sectors would be fueled by the ability to efficiently solve complex problems. Resource optimization, from energy to transportation, would become more feasible, contributing to a sustainable future [3].

In conclusion, a proof of $P = NP$ would usher in a new era of computational power with transformative effects on science, technology, and society. While challenges and uncertainties exist, the potential benefits are immense, making this a compelling area of continued research.

**Acknowledgements**

Many thanks to Sergi Simon, Jorge Félix and Melvin Vopson for their support.

**References**

[1] Stephen Arthur Cook. The P versus NP Problem, Clay Mathematics Institute. https://www.claymath.org/wp-content/uploads/2022/06/pvsnp.pdf, June 2022. Accessed September 13, 2024.

[2] Madhu Sudan. The P vs. NP problem. http://people.csail.mit.edu/madhu/papers/2010/pnp.pdf, May 2010. Accessed September 13, 2024.

[3] Lance Fortnow. The status of the P versus NP problem. *Communications of the ACM*, 52(9):78–86, 2009. doi:10.1145/1562164.1562186.

[4] Scott Aaronson. $P \stackrel{?}{=} NP$. *Open Problems in Mathematics*, pages 1–122, 2016. doi:10.1007/978-3-319-32162-2_1.

[5] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathcal{P}$ =?$\mathcal{NP}$ Question. *SIAM Journal on computing*, 4(4):431–442, 1975. doi:10.1137/0204037.

[6] Alexander A Razborov and Steven Rudich. Natural Proofs. *Journal of Computer and System Sciences*, 1(55):24–35, 1997. doi:10.1006/jcss.1997.1494.

[7] Avi Wigderson. *Mathematics and Computation: A Theory Revolutionizing Technology and Science*. Princeton University Press, 2019.

[8] Lance Fortnow. Fifty Years of P vs. NP and the Possibility of the Impossible. *Communications of the ACM*, 65(1):76–85, 2022. doi:10.1145/3460351.

[9] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[10] Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Company, 1 edition, 1979.

[11] Frank Vega. Note for the P versus NP Problem. *IPI Letters*, 2(2):14–18, Jun. 2024. URL: https://ipipublishing.org/index.php/ipil/article/view/92, doi:10.59973/ipil.92.