Note for the P versus NP problem (II)

Frank Vega [*]

Posted Date: 30 September 2024

doi: 10.20944/preprints202409.2053.v3

Keywords: Complexity classes; Boolean formula; Graph; Completeness; Polynomial time

Article

Note for the P versus NP Problem (II)

Frank Vega

Information Physics Institute, Miami, Florida, United States; vega.frank@gmail.com

Abstract: The P versus NP problem is a cornerstone of theoretical computer science, asking whether problems that are easy to check are also easy to solve. "Easy" here means solvable in polynomial time, where the computation time grows proportionally to the input size. While this problem's origins can be traced to John Nash's 1955 letter, its formalization is credited to Stephen Cook and Leonid Levin. Despite decades of research, a definitive answer remains elusive. Central to this question is the concept of NP-completeness. If even one NP-complete problem could be solved efficiently, it would imply that all problems in NP could be solved efficiently, proving P equals NP. This research proposes that EXACT CLOSURE, a notoriously difficult NP-complete problem, can be solved efficiently, thereby potentially establishing the equivalence of P and NP. This work is an expansion and refinement of the article "Note for the P versus NP problem", published in IPI Letters.

Keywords: complexity classes; Boolean formula; graph; completeness; polynomial time

MSC: Primary 68Q15; Secondary 68Q17; 68Q25

1. Introduction

The P versus NP problem is a fundamental question in computer science that asks whether problems whose solutions can be easily checked can also be easily solved [1]. “Easily” here means solvable in polynomial time, where the computation time grows proportionally to the input size [1,2]. Problems solvable in polynomial time belong to the class P, while NP includes problems whose solutions can be verified efficiently given a suitable “certificate” [2].

The central question is whether P and NP are the same. Most researchers believe that P is a strict subset of NP, meaning that some problems are inherently harder to solve than to verify. Resolving this problem has profound implications for fields like cryptography and artificial intelligence [3,4]. The P versus NP problem is widely considered one of the most challenging open questions in computer science. Techniques like relativization and natural proofs have yielded inconclusive results, suggesting the problem's difficulty [5,6]. Similar problems, such as the VP versus VNP problem in algebraic complexity, remain unsolved [7].

Resolving the P versus NP problem is often described as a “holy grail” of computer science. A positive resolution could revolutionize our understanding of computation and potentially lead to groundbreaking algorithms for critical problems. 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, 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 quickly checked using a concise proof [9].
2. **Universal Hardness:** Every problem in the class NP can be reduced to 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].
- **Exact k-Coloring Problem:** Given a graph G and a positive integer k, determine if there exists a valid coloring of G such that exactly k vertices have the same color and no adjacent vertices have the same color [11].

The provided examples represent a small subset of the extensively studied NP-complete problems relevant to our current work. A bipartite graph, denoted as $B = (U, V, E)$, is an undirected graph characterized by the existence of two node sets U, V and edges in E that only connect nodes from opposite sets.

Definition 1. ***Exact Bipartite Expansion***
INSTANCE: An n-vertex bipartite graph $B = (U, V, E)$ and positive integers b and c.
ANSWER: A b-subset of U with exactly c neighbors in W.
REMARKS: This functional problem belongs to NP-hard [12].

An independent set V' is a subset of vertices in a graph G where no two vertices in the set are connected by an edge. In addition, a vertex cover (sometimes called a node cover) of a graph G is a subset of its vertices, denoted by V', such that every edge in G has at least one endpoint in V'.

Definition 2. ***Exact Independent Vertex Cover***
INSTANCE: An undirected graph $G = (V, E)$ and a positive integer k.
QUESTION: Is there set V' of exactly k vertices such that V' is both a vertex cover and an independent set in G?
*REMARKS: Solving the **Exact Independent Vertex Cover** problem is akin to determining a 2-coloring of a bipartite graph such that one of the partitions contains exactly k vertices [11]. This task can be accomplished in polynomial time, given the efficient solvability of the 2-coloring problem in bipartite graphs [11].*

Formally, an instance of **Boolean satisfiability problem (SAT)** is a Boolean formula ϕ which is composed of:

1. Boolean variables: $x_1, x_2, \ldots, x_n$;
2. Boolean connectives: Any Boolean function with one or two inputs and one output, such as $\wedge$(AND), $\vee$(OR), $\neg$(NOT), $\Rightarrow$(implication), $\Leftrightarrow$(if and only if);
3. and parentheses.

A truth assignment for a Boolean formula ϕ is a set of values for the variables in ϕ. A satisfying truth assignment is a truth assignment that causes ϕ to be evaluated as true. A Boolean formula with a satisfying truth assignment is satisfiable. The problem SAT asks whether a given Boolean formula is satisfiable [10].

We define a CNF Boolean formula using the following terms: A literal in a Boolean formula is an occurrence of a variable or its negation [9]. A Boolean formula is in conjunctive normal form, or CNF, if it is expressed as an AND of clauses, each of which is the OR of one or more literals [9]. A Boolean formula is in 2-conjunctive normal form or 2CNF, if each clause has exactly two distinct literals [9].

For example, the Boolean formula:

$$(x_1 \vee \neg x_2) \wedge (x_3 \vee x_2) \wedge (\neg x_1 \vee \neg x_3)$$

is in $2CNF$. The first of its three clauses is $(x_1 \vee \neg x_2)$, which contains the two literals x_1 and $\neg x_2$. We define the following problem:

Definition 3. ***Exact Monotone Xor 2-satisfiability problem (EMX2SAT)***
INSTANCE: An n-variable $2CNF$ formula with monotone clauses (meaning the variables are never negated) using logic operators $\oplus$ (instead of using the operator $\vee$) and a positive integer k.
QUESTION: Is there exists a satisfying truth assignment in which exactly k of the variables are true?

Finally, we introduce the last problem:

Definition 4. ***EXACT CLOSURE***
INSTANCE: A directed graph $G = (V, A)$ and a positive integer k.
QUESTION: Is there set V' of exactly k vertices such that for all $(u, v) \in A$ either ($u \in V'$ and $v \in V'$) or ($u \notin V'$ and $v \notin V'$)?
REMARKS: We define a k-vertex set V' satisfying these conditions as a "closure".

By presenting the NP-completeness and an efficient solution to *EXACT CLOSURE*, we would establish a proof that P equals NP.

3. Main Result

This is a known result.

Theorem 1. *EXACT CLOSURE $\in NP$–complete.*

Proof. We can build an equivalent *EXACT CLOSURE* instance for any given n-vertex bipartite graph $B = (U, V, E)$ of the *Exact Bipartite Expansion* problem: To reduce the problem *Exact Bipartite Expansion* to *EXACT CLOSURE*, construct a directed graph G with two levels by duplicating each vertex in U just n^2 times and directing all edges from W to U. Given such a directed graph G, then ask for a closure set with exactly $k = b \cdot n^2 + c$ vertices. Since *Exact Bipartite Expansion* is NP-hard, and we have shown a polynomial-time reduction from the problem *Exact Bipartite Expansion* to *EXACT CLOSURE*, it follows that the functional variant of *EXACT CLOSURE* is also NP-hard. Consequently, the decision variant of *EXACT CLOSURE* is NP-complete. □

This is a main insight.

Theorem 2. *The problem EXACT CLOSURE can be reduced to $EMX2SAT$ in polynomial time.*

Proof. We can build an equivalent $EMX2SAT$ instance for any given instance $G = (V, A)$ of the *EXACT CLOSURE* problem:

1. Variables:
    - Create a variable for each vertex v in the original graph G. Denote this variable as v itself.
    - For each edge (u, v) in G, introduce two new variables denoted by x_{uv} and x_{vu}.
2. Clauses:
    - For each edge (u, v) in G, construct three clauses using the new variables:
        - $(u \oplus x_{uv})$: This enforces that either vertex u is true or the new variable x_{uv} is true (XOR).
        - $(x_{uv} \oplus v)$: This enforces that either the new variable x_{uv} is true or vertex v is true (XOR).
        - $(x_{vu} \oplus x_{uv})$: This guarantees that x_{uv} and x_{vu} have different truth values. Note that x_{vu} is not used elsewhere, so it only enforces there is exactly one true variable per each edge (u, v) over the new variables x_{uv} and x_{vu}.

Key Points about the Construction:

- The first two clauses for each edge (u, v) ensures that both variables u and v for an edge have the same truth value. This is because they represent the "state" of the edge (both in the closure or both outside). By definition, a vertex closure cannot have any outgoing edges pointing to vertices outside the closure. Therefore, no edge can exist where one vertex belongs to the solution and the other does not.
- The third clause for each edge (u, v) together ensure that exactly one of x_{uv} or x_{vu} is true in a satisfying truth assignment. Take into account this condition enforces always a true variable for each edge for every possible satisfying truth assignment.

Mapping between CLOSURE solutions and EMX2SAT assignments:

- A satisfying truth assignment in the $EMX2SAT$ formula corresponds to a valid closure of exactly k vertices in the original graph G if:
  - Vertices assigned true represent the vertices in the closure V'.
  - New variable x_{uv} assigned true represents that the corresponding edge (u, v) has both endpoints outside the closure.
  - New variable x_{vu} assigned true indicates that the corresponding edge (u, v) has both endpoints within the closure.

Why this construction works:

- The clauses enforce that a satisfying truth assignment must have consistent values for a vertex and its corresponding edge variables.
- A k $>$-vertex closure property translates to k original variables (vertices) being true in the satisfying truth assignment, along with exactly one true variable from the pair of new variables x_{uv} and x_{vu} per each edge depending on the specific closure.

Equivalence and Complexity:

- There exists a satisfying truth assignment for the $EMX2SAT$ formula with exactly $k+ \mid A \mid$ true variables if and only if there exists a closure of exactly k vertices in the original graph. ($\mid A \mid$ represents the number of edges in the graph).
- Since $EXACT\ CLOSURE$ is known to be NP-complete, this shows that $EMX2SAT$ is also NP-complete.

In essence, the proof demonstrates that solving $EMX2SAT$ is equivalent to finding a closure of exactly k vertices in the $EXACT\ CLOSURE$ problem. This implies that $EMX2SAT$ inherits the NP-completeness property from $EXACT\ CLOSURE$. □

This is the main theorem.

Theorem 3. *$EMX2SAT \in P$.*

Proof. There is a connection between finding a satisfying truth assignment in $EMX2SAT$ with exactly k true variables and finding a set of exactly k vertices that is both a vertex cover and an independent set in a specific graph construction.

Here's a breakdown of the equivalence:

1. Graph Construction:
   - Each vertex in the new graph represents a variable in the $EMX2SAT$ formula.
   - Edges are created between variables based on the structure of the $2CNF$ clauses: If two variables appear in a clause (e.g., $(x \oplus y)$), then an edge is drawn between the corresponding vertices in the graph.

2. $EMX2SAT$ and the Graph:
   - A truth assignment in $EMX2SAT$ where exactly k variables are true directly translates to a set of exactly k vertices in the constructed graph where true variables correspond to the vertices included in the set.
   - The properties of $EMX2SAT$ clauses ensure that:
     - Vertex Cover: The chosen vertices cover all the edges (due to the structure of the clauses and the way edges are formed). This satisfies the vertex cover condition.
     - Independent Set: The chosen vertices don't have any edges connecting them (because the variables are connected in the graph, and only one variable from each clause can be true). This satisfies the independent set condition.

Therefore, finding a satisfying truth assignment with exactly k true variables in $EMX2SAT$ is indeed equivalent to finding a set of exactly k vertices that fulfills both vertex cover and independent set requirements in the corresponding graph. However, we know the problem of finding a set of exactly k vertices that is both a vertex cover and an independent set can be easily solved in polynomial time [11]. Consequently, the instances of the problem $EMX2SAT$ can be solved in polynomial time as well. □

This is a key finding.

Theorem 4. $P = NP$.

Proof. A polynomial-time solution to any NP-complete problem would establish the equivalence of P and NP [9]. Given that *EXACT CLOSURE* is an NP-complete problem, a polynomial-time solution for it, as presented here, 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.

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

References

1. Cook, S.A. The P versus NP Problem, Clay Mathematics Institute. https://www.claymath.org/wp-content/uploads/2022/06/pvsnp.pdf, 2022. Accessed September 26, 2024.
2. Sudan, M. The P vs. NP problem. http://people.csail.mit.edu/madhu/papers/2010/pnp.pdf, 2010. Accessed September 26, 2024.
3. Fortnow, L. The status of the P versus NP problem. *Communications of the ACM* **2009**, *52*, 78–86. doi:10.1145/1562164.1562186.
4. Aaronson, S. $P \overset{?}{=} NP$. *Open Problems in Mathematics* **2016**, pp. 1–122. doi:10.1007/978-3-319-32162-2_1.
5. Baker, T.; Gill, J.; Solovay, R. Relativizations of the $\mathcal{P} =? \mathcal{NP}$ Question. *SIAM Journal on computing* **1975**, *4*, 431–442. doi:10.1137/0204037.
6. Razborov, A.A.; Rudich, S. Natural Proofs. *Journal of Computer and System Sciences* **1997**, *1*, 24–35. doi:10.1006/jcss.1997.1494.
7. Wigderson, A. *Mathematics and Computation: A Theory Revolutionizing Technology and Science*; Princeton University Press, 2019.
8. Fortnow, L. Fifty Years of P vs. NP and the Possibility of the Impossible. *Communications of the ACM* **2022**, *65*, 76–85. doi:10.1145/3460351.
9. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3rd ed.; The MIT Press, 2009.
10. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1 ed.; San Francisco: W. H. Freeman and Company, 1979.
11. Diestel, R. *Graph Theory*; Springer (print edition); Reinhard Diestel (eBooks), 2024.
12. Khot, S.; Saket, R. Hardness of Bipartite Expansion. Embedded Systems and Applications, 2016.