Note for the P Versus NP Problem (II)

Frank Vega

May 29, 2024

Article

Note for the P versus NP Problem (II)

Frank Vega [1]

[1] GROUPS PLUS TOURS INC., 9611 Fontainebleau Blvd, Miami, FL, 33172, USA; vega.frank@gmail.com

Abstract: P versus NP is considered as one of the most fundamental open problems in computer science. This consists in knowing the answer of the following question: Is P equal to NP? It was essentially mentioned in 1955 from a letter written by John Nash to the United States National Security Agency. However, a precise statement of the P versus NP problem was introduced independently by Stephen Cook and Leonid Levin. Since that date, all efforts to find a proof for this problem have failed. Another major complexity class is NP-complete. It is well-known that P is equal to NP under the assumption of the existence of a polynomial time algorithm for some NP-complete. We show that the Monotone Weighted-2-satisfiability problem (MW2SAT) is NP-complete and P at the same time. Certainly, we make a polynomial time reduction from every undirected graph and positive integer k in the Vertex Cover problem to an instance of MW2SAT. In this way, we show that MW2SAT is also an NP-complete problem. Moreover, we create and implement a polynomial time algorithm which decides the instances of MW2SAT. Consequently, we prove that P = NP.

Keywords: Complexity classes; boolean formula; graph; completeness; polynomial time

MSC: 68Q15; 68Q17; 68Q25

1. Introduction

The field of computer science grapples with one of its most significant and challenging unsolved problems: the P versus NP question [1]. At its core, this question asks whether efficient verification of a solution translates to efficient solving of the problem itself. Here, "efficient" refers to the existence of an algorithm that tackles the task in polynomial time, meaning the time it takes scales proportionally to the size of the input data.

The class of problems solvable by such efficient algorithms is denoted by P, or "class P." Another class, NP (standing for "nondeterministic polynomial time"), encompasses problems where solutions themselves can be verified efficiently. This verification relies on a "certificate," a piece of succinct information that quickly confirms the solution's validity [1].

The P versus NP problem essentially asks if P and NP are equivalent. If, as many believe, P is strictly contained within NP (meaning P $\neq$ NP), then some problems in NP are inherently harder to solve than to verify. This distinction would have significant ramifications for various fields like cryptography and artificial intelligence [2].

Cracking the P versus NP problem is considered a pinnacle achievement in computer science. A solution would revolutionize our understanding of computation, potentially leading to groundbreaking algorithms that address some of humanity's most pressing challenges. The difficulty of this problem is reflected in its inclusion among the Millennium Prize Problems, a prestigious set of unsolved questions offering a million-dollar reward for a correct solution [1].

2. Materials and methods

NP-complete problems are a class of computational problems that are at the heart of many important and challenging problems in computer science. They are defined by the property that they can be quickly verified, but there is no known efficient algorithm to solve them. This means that finding a solution to an NP-complete problem can be extremely time-consuming, even for relatively small inputs. In computational complexity theory, a problem is considered NP-complete if it meets the following two criteria:

1. **Membership in NP**: A solution to an NP-complete problem can be verified in polynomial time. This means that there is an algorithm that can quickly check whether a proposed solution is correct [3].
2. **Reduction to NP-complete problems**: Any problem in NP can be reduced to an NP-complete problem in polynomial time. This means that any NP-problem can be transformed into an NP-complete problem by making a small number of changes [3].

If it were possible to find an efficient algorithm for solving any one NP-complete problem, then this algorithm could be used to solve all NP problems in polynomial time. This would have a profound impact on many fields, including cryptography, artificial intelligence, and operations research [2]. Here are some examples of NP-complete problems:

- **Boolean satisfiability problem (SAT)**: Given a Boolean formula, determine whether there is an assignment of truth values to the variables that makes the formula true [4].
- **Vertex Cover problem**: Given an undirected graph $G = (V, E)$ (V is the set of vertices and E is the set of edges) and positive integer k, determine whether there is a set $V' \subseteq V$ of at most k vertices such that for each edge $(u, v) \in E$ at least one of u and v belongs to V' [4].

These are just a few examples of the many NP-complete problems that have been studied and have a close relation with our current result. In addition, an edge cover of a graph G is a subset of its edges, denoted by E', such that every vertex in G belongs to at least one edge in E'.

Definition 1. ***Minimum Edge Cover***
INSTANCE: An undirected graph $G = (V, E)$.
ANSWER: Find the smallest subset $E' \subseteq E$ such that for every vertex $v \in V$ at least one edge $e \in E'$ contains v as an endpoint?
REMARKS: This problem can be easily solved in polynomial time by graph matching [4].

In this work, we show there is an NP-complete problem that can be solved in polynomial time using the previous problem. Consequently, we prove that P is equal to NP.

3. Results

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 [4].

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 [3]. 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 [3]. A Boolean formula is in 2-conjunctive normal form or $2CNF$, if each clause has exactly two distinct literals [3].

For example, the Boolean formula:

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

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

We define the following problem:

Definition 2. ***Monotone Weighted-2-satisfiability problem (MW2SAT)***

INSTANCE: An n-variable $2CNF$ *formula with monotone clauses (meaning the variables are never negated) and a positive integer k.*

QUESTION: Is there exists a satisfying truth assignment in which at most k of the variables are true?

REMARKS: The general case (i.e. whenever it is not monotone) is well-known that belongs to NP-complete [5].

The following is key Lemma.

Lemma 1. $MW2SAT \in NP$*–complete.*

Proof. We can prove that the Vertex Cover problem can be entirely expressed within the MW2SAT framework. Here's a breakdown:

1. **Vertex Cover as MW2SAT**: We can transform any Vertex Cover instance into a MW2SAT problem. Imagine a graph with an associated Vertex Cover problem.
2. **Variable Assignment**: Create a Boolean variable for each vertex in the graph. A variable being "true" signifies the vertex is included in the potential solution (vertex cover).
3. **Clause Construction**: For every edge connecting vertices u and v, construct a MW2SAT clause $(u \vee v)$. This clause ensures at least one of the connected vertices (u or v) must be included (true) in the solution for the clause to be satisfied.
4. **Solution Equivalence**: A satisfying assignment for the generated MW2SAT formula directly corresponds to a solution for the original Vertex Cover problem. Each satisfied clause guarantees an edge is covered by at least one vertex in the assigned true variables (chosen vertex cover).
5. **Optimality**: The MW2SAT solution with at most k true variables translates to a Vertex Cover with at most k vertices, fulfilling the requirement of minimizing the number of vertices in the cover. Conversely, a k-vertex Vertex Cover solution can be mapped back to a MW2SAT solution with k true variables.
6. **Shared Complexity**: Since Vertex Cover is NP-complete, and we've shown it can be reduced to MW2SAT, it implies MW2SAT is also NP-complete. In simpler terms, if solving Vertex Cover is inherently difficult (NP-complete), then solving MW2SAT, which can express Vertex Cover problems, must be at least as hard.

In essence, this proof establishes MW2SAT as a more general framework that can encompass problems like Vertex Cover. The equivalence between solutions and the preservation of NP-completeness solidify this connection. □

This is the main theorem.

Theorem 1. $MW2SAT \in P$.

Proof. There is a connection between finding a satisfying truth assignment in MW2SAT with at most k true variables and finding a set of at most k vertices associated to an edge cover in a specific graph construction. Here's a breakdown of the equivalence:

1. Graph Construction:
   - Each vertex in the original graph G represents a variable in the MW2SAT 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 G.
2. MW2SAT and the Line Graph: We create a line graph L of the original graph G which has a node for each edge in G and an edge joining those nodes if the two edges in G share a common node [6].

- A truth assignment in MW2SAT where at most k variables are true directly translates to a set of at most k common nodes from the edges in a minimum edge cover of the line graph L where true variables correspond to the vertices included in the set.
- The properties of MW2SAT clauses ensure that:
  - Vertex Cover: The chosen vertices cover all the edges (due to the structure of the clauses and the way common nodes from edges are formed in L). This satisfies the vertex cover condition in G since it inherently satisfies the edge cover condition in L.
  - Minimum Cover: The chosen common vertices associated to the edges in the line graph L satisfy the minimum cover set condition when we apply the Minimum Edge Cover algorithm to the line graph L.

Therefore, finding a satisfying truth assignment with at most k true variables in MW2SAT is indeed equivalent to finding a set of at most k common vertices that fulfills each edge inside of a minimum edge cover in the line graph L. However, we know the problem of finding the smallest edge cover in an undirected graph can be easily solved in polynomial time [4]. Consequently, the instances of the problem MW2SAT can be solved in polynomial time as well. $\square$

4. Discussion

The **Minimum Edge Cover** problem is equivalent to solving the graph matching problem [4]. Indeed, there are more than one algorithm that feasibly solves this problem. For example, we have the Hopcroft-Karp algorithm on bipartite graph [7]. We create a software programming implementation in Python for the whole algorithm that solves MW2SAT instances just using the NetworkX's Library and its dependencies (this code in Python would be outside of the necessary correctness of the paper and thus, this can only be considered as an appendix that will not compromise the whole result). [8]. This is placed into a GitHub repository under my GitHub username "frankvegadelgado" [8]. The last commit was on March 29th of 2024 with a SHA commit **65ca51749ee391adcb378ab32cbe2434b486a1a1** [8].

5. Conclusion

A proof of $P = NP$ will have stunning practical consequences, because it possibly leads to efficient methods for solving some of the important problems in computer science [1]. The consequences, both positive and negative, arise since various NP-complete problems are fundamental in many fields [2]. But such changes may pale in significance compared to the revolution an efficient method for solving NP-complete problems will cause in mathematics itself [1]. Research mathematicians spend their careers trying to prove theorems, and some proofs have taken decades or even centuries to be discovered after problems have been stated [1]. A method that guarantees to find proofs for theorems, should one exist of a "reasonable" size, would essentially end this struggle [1].

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 27 March 2024.
2. Fortnow, L. The status of the P versus NP problem. *Communications of the ACM* **2009**, *52*, 78–86. https://doi.org/10.1145/1562164.1562186.
3. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3rd ed.; The MIT Press, 2009.
4. 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.
5. Flum, J.; Grohe, M. *Parameterized Complexity Theory, Springer*; Heidelberg, 2006; pp. 69–70. https://doi.org/10.1007/3-540-29953-X.
6. Hemminger, R.L. Line graphs and line digraphs. *Selected Topics in Graph Theory* **1983**, pp. 271–305.
7. Hopcroft, J.E.; Karp, R.M. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing* **1973**, 2, 225–231. https://doi.org/10.1137/0202019.
8. Vega, F. MENTE | MW2SAT Solver. https://github.com/frankvegadelgado/mente, 2024. Accessed 29 March 2024.

Short Biography of Authors

Frank Vega is essentially a Back-End Programmer and Mathematical Hobbyist who graduated in Computer Science in 2007. In May 2022, The Ramanujan Journal accepted his mathematical article about the Riemann hypothesis. The article "Robin's criterion on divisibility" makes several significant contributions to the field of number theory. It provides a proof of the Robin inequality for a large class of integers, and it suggests new directions for research in the area of analytic number theory.