Article

Note for the P versus NP Problem

Frank Vega *

Posted Date: 14 March 2024

doi: 10.20944/preprints201908.0037.v10

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

 doi:10.20944/preprints201908.0037.v10

Article

Note for the P versus NP Problem

Frank Vega

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 Xor 2-satisfiability problem (MWX2SAT) is NP-complete and P at the same time. Certainly, we make a polynomial time reduction from every directed graph and positive integer k in the K-CLOSURE problem to an instance of MWX2SAT. In this way, we show that MWX2SAT is also an NP-complete problem. Moreover, we create and implement a polynomial time algorithm which decides the instances of MWX2SAT. Consequently, we prove that P = NP.

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

MSC: 68Q15; 68Q17; 68Q25

1. Introduction

P versus NP is one of the most important and challenging problems in computer science [1]. It asks whether every problem whose solution can be quickly verified can also be quickly solved. The informal term "quickly" here refers to the existence of an algorithm that can solve the task in polynomial time [1]. The general class of problems for which such an algorithm exists is called P or "class P" [1].

Another class of problems called NP, which stands for "nondeterministic polynomial time", is defined by the property that if an input to a problem is a solution, then it can be quickly verified [1]. The P versus NP problem asks whether P equals NP. If it turns out that $P \neq NP$, which is widely believed to be the case, it would mean that there are problems in NP that are harder to compute than to verify [1]. This would have profound implications for various fields, including cryptography and artificial intelligence [2].

Solving the P versus NP problem is considered to be one of the greatest challenges in computer science [1]. A solution would have a profound impact on our understanding of computation and could lead to the development of new algorithms and techniques that could solve many of the world's most pressing problems [1]. The problem is so difficult that it is considered to be one of the seven Millennium Prize Problems, which are a set of seven unsolved problems that have been offered a 1 million prize 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:

 doi:10.20944/preprints201908.0037.v10

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].
- **K-CLOSURE problem**: Given a directed graph $G = (V, A)$ (V is the set of vertices and A is the set of edges) and positive integer k, determine whether there is a set V' of at most k vertices such that for all $(u, v) \in A$ either $u \in V'$ or $v \notin V'$ (see reference [Queyranne, 1976] from the Johnson and Garey book) [4]. Note that in this problem the statement "either $u \in V'$ or $v \notin V'$" does mean the same as: ($u \in V'$ or $v \in V'$) or ($u \notin V'$ or $v \notin V'$) since the logical implication of the word "**Either**" indicates that at least one of the following statements must be true, but not necessarily both.

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. On the one hand, 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'. On the other hand, 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.

Definition 1. ***Vertex Cover and Independent Set***

INSTANCE: An undirected graph $G = (V, E)$ and a positive integer k.

QUESTION: Is there set V' of at most k vertices such that V' is both a vertex cover and an independent set in G?

REMARKS: This problem can be easily solved in polynomial time [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,

 doi:10.20944/preprints201908.0037.v10

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 Xor 2-satisfiability problem (MWX2SAT)***

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 at most k of the variables are true?

The following is key Lemma.

Lemma 1. $MWX2SAT \in NP$*–complete.*

Proof. For any given instance $G = (V, A)$ of the *K-CLOSURE* problem, one can construct an equivalent $MWX2SAT$ problem with a variable for each vertex of a graph and two variables for each edge of a graph. Each edge (u, v) of the graph may be represented by the $2CNF$ clauses $(u \oplus x_{uv}) \wedge (x_{uv} \oplus v) \wedge (x_{vu} \oplus x_{uv})$ where x_{vu} and x_{uv} are two new variables such that for a possible satisfying truth assignment, either both variables u and v are true and belong to a closure V' or both variables u and v are false and belong to $V - V'$. By definition, the k-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. Both endpoints of any edge must either be inside the closure or outside it. Then the satisfying instances of the resulting $2CNF$ formula using logic operators $\oplus$ encode solutions to the *K-CLOSURE* problem, and there is a satisfying truth assignment with at most $k + |A|$ true variables if and only if there is a closure with at most k vertices where $|\ldots|$ is the cardinality set function. Therefore, like *K-CLOSURE*, $MWX2SAT$ is NP-complete. □

This is the main insight.

Theorem 1. $MWX2SAT \in P$.

Proof. There is a connection between finding a satisfying truth assignment in $MWX2SAT$ with at most k true variables and finding a set of at most 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 original graph represents a variable in the $MWX2SAT$ 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. $MWX2SAT$ and the Graph:
   - A truth assignment in $MWX2SAT$ where at most k variables are true directly translates to a set of at most k vertices in the constructed graph where true variables correspond to the vertices included in the set.
   - The properties of $MWX2SAT$ clauses ensure that:

 doi:10.20944/preprints201908.0037.v10

- 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 at most k true variables in $MWX2SAT$ is indeed equivalent to finding a set of at most 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 at most k vertices that is both a vertex cover and an independent set can be easily solved in polynomial time [4]. Consequently, the instances of the problem $MWX2SAT$ can be solved in polynomial time as well. □

This is the main theorem.

Theorem 2. *There is a quadratic polynomial time algorithm for the problem $MWX2SAT$.*

Proof. Suppose we have the following sequence of variables in a given instance of $MWX2SAT$:

$$x_1, \ldots, x_n.$$

For each variable x_i in the $2CNF$ formula, we define the functions f and g as,

- $f(x_i)$ is the number of variables x_j such that either $(x_i \vee x_j)$ or $(x_j \vee x_i)$ belongs to the $2CNF$ formula whenever $j > i$;
- $g(x_i)$ is the number of variables x_j such that either $(x_i \vee x_j)$ or $(x_j \vee x_i)$ belongs to the $2CNF$ formula whenever $j < i$.

We define a state as a quadruple (i, s, r, t) of integers. This state represents the fact that,

$$\text{"}\textit{the subset of variables } x_1, \ldots, x_i$$

$$\textit{with } s \textit{ true variables}$$

$$\textit{where } -m \leq r \leq m \textit{ and } -m \leq t \leq m''\text{,}$$

where m is the amount of clauses into the $2CNF$ formula. Each state (i, s, r, t) has two next states:

1. $(i+1, s+1, r+f(x_{i+1}), t-g(x_{i+1}))$, implying that x_{i+1} is included in the subset and it is evaluated as true;
2. $(i+1, s, r-g(x_{i+1}), t+f(x_{i+1}))$, implying that x_{i+1} is included in the subset and it is evaluated as false.

Starting from the initial state $(0,0,0,0)$, it is possible to use any graph search algorithm (e.g. **Breadth-first search (BFS)** [3]) to search any state $(n, \varpi, 0, 0)$ such that $0 < \varpi \leq k$. Certainly, we satisfy all the clauses if they contain exactly one true literal just adding 1 by the true literal from the left most position (otherwise adding 1 by the false literal from the left most position) and subtracting 1 by the false literal from the right most position (otherwise subtracting 1 by the true literal from the right most position). The run-time of this algorithm is at most linear in the number of states. The number of states is at bounded by $n^2 \cdot 4 \cdot m^2$ times and therefore, the whole time required is $O((n \cdot m)^2)$. We create our software programming implementation in Python for this algorithm [5]. This is placed into a GitHub repository under my GitHub username "frankvegadelgado" [5]. The last commit was on February 27th of 2024 with a SHA commit **eec2ccb2bba51e3efaf3ad7d722b948b88861ea9** [5]. □

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 1 March 2024.

 doi:10.20944/preprints201908.0037.v10

2. Fortnow, L. The status of the P versus NP problem. *Communications of the ACM* **2009**, *52*, 78–86. doi: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. Vega, F. ALMA | MWX2SAT Solver. https://github.com/frankvegadelgado/alma, 2024. Accessed 1 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.