P vs NP

FRANK VEGA, Joysonic, Serbia

P versus NP is considered as one of the most important open problems in computer science. This consists in knowing the answer of the following question: Is P equal to NP? It is one of the seven Millennium Prize Problems selected by the Clay Mathematics Institute. This question was first mentioned in a letter written by John Nash to the National Security Agency in 1955. A precise statement of the P versus NP problem was introduced independently in 1971 by Stephen Cook and Leonid Levin. Since that date, all efforts to find a proof for this problem have failed. To attack the P versus NP question the concept of NP-completeness has been very useful. If any single NP-complete problem can be solved in polynomial time, then every NP problem has a polynomial time algorithm. MONOTONE 3SAT is a known NP-complete problem. We prove MONOTONE 3SAT is in P. In this way, we demonstrate the P versus NP problem.

Additional Key Words and Phrases: Complexity Classes, Completeness, Polynomial Time, 3SAT, Quadratic Residue

1 THEORY

An important complexity class is *NP–complete* [7]. A language $L \subseteq \{0,1\}^*$ is *NP–complete* if

- $L \in NP$, and
- $L' \leq_p L$ for every $L' \in NP$.

If L is a language such that $L' \leq_p L$ for some $L' \in$ *NP–complete*, then L is *NP–hard* [5]. Moreover, if $L \in NP$, then $L \in$ *NP–complete* [5]. A Boolean formula ϕ 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 formula with a satisfying truth assignment is a satisfiable formula. 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 [5]. 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 [5]. A Boolean formula is in 3-conjunctive normal form or $3CNF$, if each clause has exactly three distinct literals [5].

For example, the Boolean formula

$$(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

is in $3CNF$. The first of its three clauses is $(x_1 \vee \neg x_1 \vee \neg x_2)$, which contains the three literals x_1, $\neg x_1$, and $\neg x_2$. A relevant *NP–complete* language is $3CNF$ satisfiability, or $3SAT$ [5]. In $3SAT$, it is asked whether a given Boolean formula ϕ in $3CNF$ is satisfiable. This problem remains in *NP–complete* even if each clause contains either only negated or un-negated variables (*MONOTONE 3SAT*) [7].

In number theory, an integer q is called a quadratic residue modulo n if it is congruent to a perfect square modulo n [8]; i.e., if there exists an integer x such that:

$$x^2 \equiv q (mod\ n).$$

Otherwise, q is called a quadratic nonresidue modulo n [8]. When in the context is clear the terminology "quadratic residue" and "quadratic nonresidue", then it is dropped the adjective

Author's address: Frank Vega, Joysonic, Uzun Mirkova 5, Belgrade, 11000, Serbia, vega.frank@gmail.com.

"quadratic" [8]. We use the shorthand notations $q\ R\ p$ and $q\ N\ p$, to indicated that q is a quadratic residue or nonresidue, respectively. [8].

2 RESULTS

Theorem 2.1. *MONOTONE 3SAT* $\in P$.

Proof. Let ϕ be a Boolean formula in $3CNF$ of n variables with m clauses. Let $p_1, p_2, \ldots, p_n$ be the first n odd primes such that they have 2 as a quadratic nonresidue. We assign for each variable in the Boolean formula ϕ a unique of these prime numbers such that the variable x_1 will have the prime p_1 and so on consecutively.

We shall say z satisfies ϕ if the assignment $(z\ R\ p_1, z\ R\ p_2, \ldots, z\ R\ p_n)$ satisfies ϕ. This means in a satisfying truth assignment T the variable x_1 is true if $z\ R\ p_1$ or x_2 is false when $z\ N\ p_2$ and so forth. Now, for each clause c_k in ϕ we construct an expression of nonresidues and residues that make the clause false for a possible candidate z. For example, in the clause $c_k = (x_r \vee x_s \vee x_t)$ for $1 \leq r, s, t \leq n$, then a solution of the simultaneous nonresidues $z\ N\ p_r$, $z\ N\ p_s$ and $z\ N\ p_t$ guarantee the clause will be false because x_r would be false, x_s would be false and x_t would be false. However, we already know that when $z\ N\ p_r$, $z\ N\ p_s$ and $z\ N\ p_t$, then $(2 \times z)\ R\ p_r$, $(2 \times z)\ R\ p_s$ and $(2 \times z)\ R\ p_t$ because 2 is a nonresidue modulo every of these chosen primes and the multiplication of a nonresidue with a nonresidue is a residue [8]. Since p_r, p_s and p_t are primes, then we can assure that $(2 \times z)\ R\ (p_r \times p_s \times p_t)$ due to the following property: a number x is a residue modulo y when x is a residue modulo for every prime power dividing y [8].

Therefore, when $(2 \times z)\ R\ (p_r \times p_s \times p_t)$, then we guarantee the clause c_k will be evaluated as false. In contraposition, in the clause $c_{k'} = (\neg\ x_r \vee \neg\ x_s \vee \neg\ x_t)$ for $1 \leq r, s, t \leq n$, then a solution of the simultaneous residues $z\ R\ p_r$, $z\ R\ p_s$ and $z\ R\ p_t$ guarantee the clause will be false because x_r would be true, x_s would be true and x_t would be true. Since p_r, p_s and p_t are primes, then we can assure that $z\ R\ (p_r \times p_s \times p_t)$ due to the property mentioned above. Hence, when $z\ R\ (p_r \times p_s \times p_t)$, then we guarantee the clause $c_{k'}$ will be evaluated as false. Consequently, if we guarantee that some number z complies with $(2 \times z)\ N\ (p_r \times p_s \times p_t)$ or $z\ N\ (p_r \times p_s \times p_t)$ for every clause c in ϕ, then z will correspond to a satisfying truth assignment for ϕ.

However, it is enough to search an odd or even number z between all the values of $0 \leq z \leq p_{n-2} \times p_{n-1} \times p_n$ such that z and $2 \times z$ complies with the nonresidues of the triple multiplication of primes from each clause due to the following property: a number x is a nonresidue modulo y when x is a nonresidue modulo for at least one prime power dividing y [8]. Certainly, for every triple of primes p_i, p_j and p_k, within the numbers between 0 and $p_i \times p_j \times p_k$ there are the fully combinations of residues and nonresidues between the primes p_i, p_j and p_k [13]. When we mean combinations, we actually mean for example a number that is residue from p_i and p_j, but it is nonresidue with p_k and so forth. In this way, $p_{n-2} \times p_{n-1} \times p_n$ is the greatest upper bound which complies that property in ϕ. Thus, *MONOTONE 3SAT* $\in P$. Certainly, we can find the first n odd primes such that they have 2 as a quadratic nonresidue just checking for every odd prime p whether

$$p \equiv 3 (mod\ 8)$$

or

$$p \equiv 5 (mod\ 8)$$

as a consequence of the Euler's criterion [13]. Indeed, there are infinitely many primes of the form $8 \times k + 3$ or $8 \times k + 5$ due to Dirichlet's theorem on arithmetic progressions [14]. Moreover, the n^{th} odd prime which has 2 as a quadratic nonresidue is polynomially bounded by $n \times \ln n$, because of $\pi(x; 8, 3)$ and $\pi(x; 8, 5)$ are asymptotic to $\frac{Li(x)}{\varphi(8)}$ where φ is the Euler's totient function and Li is the Eulerian logarithmic integral [14]. Consequently, the search of the number z through the iteration from 0 to $p_{n-2} \times p_{n-1} \times p_n$ can be done $O(n^4)$. In addition, we can feasible search these special first primes, because we can make the primality test of a number in polynomial time [2]. □

Theorem 2.2. $P = NP$.

Proof. If any single *NP–complete* problem can be solved in polynomial time, then $P = NP$ [12]. As a consequence of Theorem 2.1, the answer of the P versus NP problem will be $P = NP$. □

3 DISCUSSION

Cryptography, for example, relies on certain problems being difficult. A constructive and efficient solution to an *NP–complete* problem such as $3SAT$ will break most existing cryptosystems including: Public-key cryptography [10], symmetric ciphers [11] and one-way functions used in cryptographic hashing [6]. These would need to be modified or replaced by information-theoretically secure solutions not inherently based on *P–NP* equivalence.

There are enormous consequences that will follow from rendering tractable many currently mathematically intractable problems. For instance, many problems in operations research are *NP–complete*, such as some types of integer programming and the traveling salesman problem [9]. Efficient solutions to these problems have enormous implications for logistics [4]. Many other important problems, such as some problems in protein structure prediction, are also *NP–complete*, so this will spur considerable advances in biology [3].

Indeed, a proof of $P = NP$ could solve not merely one Millennium Problem but all seven of them [1]. This observation is based on once we fix a formal system such as the first-order logic plus the axioms of ZF set theory, then we can find a demonstration in time polynomial in n when a given statement has a proof with at most n symbols long in that system [1]. This is assuming that the other six Clay conjectures have ZF proofs that are not too large such as it was the Perelman's case [1].

REFERENCES

[1] Scott Aaronson. 2017. P $\stackrel{?}{=}$ NP. *Electronic Colloquium on Computational Complexity, Report No. 4* (2017).

[2] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. 2004. PRIMES is in P. *Annals of Mathematics* 160, 2 (2004), 781–793. https://doi.org/10.4007/annals.2004.160.781

[3] Bonnie Berger and Tom Leighton. 1998. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology* 5, 1 (1998), 27–40. https://doi.org/10.1145/279069.279080

[4] Stephen A. Cook. 2000. The P versus NP Problem. (April 2000). At http://www.claymath.org/sites/default/files/pvsnp.pdf.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms* (3 ed.). The MIT Press.

[6] Debapratim De, Abishek Kumarasubramanian, and Ramarathnam Venkatesan. 2007. Inversion Attacks on Secure Hash Functions Using sat Solvers. In *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 377–382. https://doi.org/10.1007/978-3-540-72788-0_36

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

[8] Carl Friedrich Gauss and Arthur A. Clarke. 1986. *Disquisitiones Arithemeticae* (2 ed.). New York: Springer.

[9] Oded Goldreich. 2010. *P, NP, and NP-Completeness: The basics of computational complexity*. Cambridge University Press.

[10] Satoshi Horie and Osamu Watanabe. 1997. Hard instance generation for SAT. *Algorithms and Computation* (1997), 22–31. https://doi.org/10.1007/3-540-63890-3_4

[11] Fabio Massacci and Laura Marraro. 2000. Logical Cryptanalysis as a SAT Problem. *Journal of Automated Reasoning* 24, 1 (2000), 165–203. https://doi.org/10.1023/A:1006326723002

[12] Christos H. Papadimitriou. 1994. *Computational complexity*. Addison-Wesley.

[13] Joseph H. Silverman. 2012. *A Friendly Introduction to Number Theory* (4 ed.). Pearson Education, Inc.

[14] Arnold Walfisz. 1936. Zur additiven Zahlentheorie. II. *Mathematische Zeitschrift* 40, 1 (1936), 592–607. https://doi.org/10.1007/BF01218882