MONOTONE-MIN-3SAT in Polynomial Time: A Proof of P = NP

Frank Vega *

Posted Date: 8 December 2025

doi: 10.20944/preprints202409.2053.v20

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

 doi:10.20944/preprints202409.2053.v20

Article

MONOTONE-MIN-3SAT in Polynomial Time: A Proof of P = NP

Frank Vega

Information Physics Institute, 840 W 67th St, Hialeah, FL 33012, USA; 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 NP problems could be solved efficiently, proving P=NP. This research proposes a groundbreaking claim: MONOTONE-MIN-3SAT can be solved in polynomial time and belongs to NP-complete, establishing the equivalence of P and NP.

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

MSC: 68Q15, 68Q17, 05C69, 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" [1,2]. Alternatively, P and NP can be defined in terms of deterministic and non-deterministic Turing machines with polynomial-time complexity [1,2].

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].

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 [3]. A polynomial-time algorithm for any NP-complete problem would directly imply P equals NP [8]. Our work focuses on presenting such an algorithm for a new NP-complete problem.

 doi:10.20944/preprints202409.2053.v20

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 [8].
2. **Universal Hardness:** Every problem in the class NP can be reduced to this problem without significant computational overhead [8].

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) Problem:** Given a logical expression in conjunctive normal form, determine if there exists an assignment of truth values to its variables that makes the entire expression true [9].
- **Boolean 3-Satisfiability (3SAT) Problem:** Given a Boolean formula in conjunctive normal form with exactly three literals per clause, determine if there exists a truth assignment to its variables that makes the formula evaluate to true [9].

The provided examples represent a small subset of the extensively studied NP-complete problems relevant to our current work. The satisfiability problem (SAT) is one of the most fundamental problems in computational complexity theory. A Boolean formula ϕ is built from:

1. Boolean variables $x_1, x_2, \ldots, x_n$;
2. Boolean connectives, i.e., Boolean functions with one or two inputs and one output, such as $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT), $\Rightarrow$ (implication), and $\Leftrightarrow$ (if and only if);
3. Parentheses, used to indicate the structure of the formula.

A truth assignment for ϕ is a mapping from the variables of ϕ to the Boolean values $\{true, false\}$. A truth assignment is *satisfying* if it makes ϕ evaluate to true. A Boolean formula is *satisfiable* if it has at least one satisfying truth assignment. A *literal* is a Boolean variable or its negation. A Boolean formula is in conjunctive normal form (CNF) if it is a conjunction (AND) of clauses, where each clause is a disjunction (OR) of one or more literals [8]. The SAT problem asks whether a given Boolean formula in CNF is satisfiable [9]. A 3CNF formula is a CNF formula in which each clause contains exactly three distinct literals [8]. For example, the following formula is in 3CNF:

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

The first clause $(x_1 \vee \neg x_2 \vee x_3)$ contains the three literals x_1, $\neg x_2$, and x_3. The restriction of SAT to formulas in 3CNF is called 3SAT [8].

While the classical satisfiability problem seeks to maximize the number of satisfied clauses (or determine if all clauses can be satisfied), the *minimum satisfiability* problem considers the complementary optimization goal: finding a truth assignment that satisfies as few clauses as possible. Kohli, Krishnamurti, and Mirchandani [10] introduced the minimum satisfiability problem (MINSAT) and proved that it is NP-hard, even when restricted to formulas where each clause contains at most two literals (MIN-2SAT). Their result established that minimization variants of satisfiability are computationally intractable, contrasting with the polynomial-time solvability of certain restricted satisfiability problems.

In this work, we consider a further restriction that combines two structural constraints: monotonicity and bounded clause size. Monotone SAT variants, where each clause contains only positive literals or only negative literals, have been extensively studied [11]. These restrictions arise naturally in various applications and exhibit distinct computational properties from their non-monotone counterparts.

 doi:10.20944/preprints202409.2053.v20

We now formalize the variant of interest.

Definition 1 (MONOTONE-MIN-3SAT)**.**
Instance: *A Boolean formula ϕ in 3CNF such that every clause is monotone (each clause contains either only positive literals or only negative literals) and has three distinct literals, together with a positive integer k.*
Question: *Does there exist a truth assignment that satisfies at most k clauses?*

By presenting the NP-completeness and a polynomial-time solution to MONOTONE-MIN-3SAT, we would establish a proof that P equals NP.

3. Main Result

Our main result establishes the computational complexity of this problem.

Theorem 1. MONOTONE-MIN-3SAT *is NP-complete.*

Proof. We first show that MONOTONE-MIN-3SAT is in NP. Given a truth assignment for the variables in ϕ, we can verify in polynomial time that at most k clauses are satisfied by evaluating each clause under the assignment and counting the number of satisfied clauses. This verification can be done in $O(n \cdot m)$ time, where n is the number of variables and m is the number of clauses.

To show NP-hardness, we reduce from MIN-2SAT, which is NP-complete [10]. Let (ψ, k') be an instance of MIN-2SAT, where ψ is a Boolean formula in 2CNF over variables $V = \{x_1, x_2, \ldots, x_n\}$ and clauses $C = \{c_1, c_2, \ldots, c_m\}$, and k' is a positive integer. Each clause c_i has exactly two literals and contains at most one unnegated variable [10].

We construct an instance (ϕ, k) of MONOTONE-MIN-3SAT as follows.

Construction

For each clause $c_i = (\neg\ell_{i,1} \vee \neg\ell_{i,2})$ in ψ, introduce fresh variables y_i, z_i, w_i (unique to c_i), and define three clauses in ϕ:

$$\begin{aligned} d_{i,1} &= (\neg\ell_{i,1} \vee \neg y_i \vee \neg z_i), \\ d_{i,2} &= (\neg\ell_{i,2} \vee \neg y_i \vee \neg w_i), \\ d_{i,3} &= (y_i \vee z_i \vee w_i). \end{aligned}$$

For each clause $c_i = (\ell_{i,1} \vee \neg\ell_{i,2})$ in ψ, introduce fresh variables y_i, z_i, w_i, u_i, v_i (unique to c_i), and define three clauses in ϕ:

$$\begin{aligned} d_{i,1} &= (\ell_{i,1} \vee y_i \vee z_i), \\ d_{i,2} &= (\neg\ell_{i,2} \vee \neg w_i \vee \neg u_i), \\ d_{i,3} &= (y_i \vee w_i \vee v_i). \end{aligned}$$

Set

$$k \;=\; m + k'.$$

Gadget Behavior

Fix an assignment τ on V, and consider a single gadget for c_i:

- If $c_i = (\neg\ell_{i,1} \vee \neg\ell_{i,2})$ is unsatisfied by τ (i.e., both $\neg\ell_{i,1}$ and $\neg\ell_{i,2}$ are false under τ), then by setting $y_i = z_i = w_i =$ true, $d_{i,3}$ is satisfied and $d_{i,1}, d_{i,2}$ are unsatisfied; thus exactly one clause is satisfied. Moreover, regardless of how y_i, z_i, w_i are set, at least one of $d_{i,1}, d_{i,2}, d_{i,3}$ is satisfied, so the minimum achievable satisfied count for this gadget is 1.
- If $c_i = (\neg\ell_{i,1} \vee \neg\ell_{i,2})$ is satisfied by τ (i.e., at least one of $\neg\ell_{i,1}, \neg\ell_{i,2}$ is true under τ), then by setting $y_i = z_i = w_i =$ false, $d_{i,1}$ and $d_{i,2}$ are satisfied and $d_{i,3}$ is unsatisfied; thus exactly two clauses

 doi:10.20944/preprints202409.2053.v20

are satisfied. Furthermore, no assignment to y_i, z_i, w_i can make fewer than two of $d_{i,1}, d_{i,2}, d_{i,3}$ satisfied: if $y_i = z_i = w_i =$ true, then $d_{i,3}$ is satisfied and at least one of $d_{i,1}, d_{i,2}$ is satisfied because at least one of $\neg\ell_{i,1}, \neg\ell_{i,2}$ is true. Hence the minimum achievable satisfied count for this gadget is 2.

- If $c_i = (\ell_{i,1} \vee \neg\ell_{i,2})$ is unsatisfied by τ (i.e., both $\ell_{i,1}$ and $\neg\ell_{i,2}$ are false under τ), then by setting $y_i = z_i = v_i =$ false; $w_i = u_i =$ true, $d_{i,3}$ is satisfied and $d_{i,1}, d_{i,2}$ are unsatisfied; thus exactly one clause is satisfied. Moreover, regardless of how y_i, z_i, w_i, u_i, v_i are set, at least one of $d_{i,1}, d_{i,2}, d_{i,3}$ is satisfied, so the minimum achievable satisfied count for this gadget is 1.
- If $c_i = (\ell_{i,1} \vee \neg\ell_{i,2})$ is satisfied by τ (i.e., at least one of $\ell_{i,1}, \neg\ell_{i,2}$ is true under τ), then by setting $y_i = w_i = v_i =$ false; $z_i = u_i =$ true, $d_{i,1}$ and $d_{i,2}$ are satisfied and $d_{i,3}$ is unsatisfied; thus exactly two clauses are satisfied. Furthermore, no assignment to y_i, z_i, w_i, u_i, v_i can make fewer than two of $d_{i,1}, d_{i,2}, d_{i,3}$ satisfied: if $y_i = z_i = v_i =$ false; $w_i = u_i =$ true, then $d_{i,3}$ is satisfied and at least one of $d_{i,1}, d_{i,2}$ is satisfied because at least one of $\ell_{i,1}, \neg\ell_{i,2}$ is true. Hence the minimum achievable satisfied count for this gadget is 2.

Therefore, for any fixed assignment on V that satisfies exactly s clauses of ψ, there exists an extension to the auxiliary variables attaining exactly

$$(m - s) \cdot 1 \; + \; s \cdot 2 \; = \; m + s$$

satisfied clauses in ϕ, and no extension can attain fewer than $m + s$.

Correctness

We prove the reduction is correct in both directions.

($\Rightarrow$)

Suppose ψ has an assignment τ satisfying at most k' clauses (i.e., $s \leq k'$). By the gadget behavior, there is an extension τ' to the auxiliary variables that satisfies exactly $m + s \leq m + k' = k$ clauses of ϕ. Thus (ϕ, k) is a yes-instance of MONOTONE-MIN-3SAT.

($\Leftarrow$)

Conversely, suppose (ϕ, k) is a yes-instance, i.e., there exists an assignment τ' satisfying at most $k = m + k'$ clauses of ϕ. Let τ be the restriction of τ' to V, and let s be the number of clauses of ψ satisfied by τ. By the gadget lower bound, every extension of τ satisfies at least $m + s$ clauses of ϕ, so $m + s \leq k = m + k'$. Therefore $s \leq k'$, and ψ is a yes-instance of MIN-2SAT.

Conclusions

The reduction is polynomial: it introduces a constant number of fresh variables and clauses per original clause. By the two-directional correctness and membership in NP, MONOTONE-MIN-3SAT is NP-complete. □

This is a main insight.

Theorem 2 (Main Insight)**.** *Maximizing the number of unsatisfied clauses in a monotone 2CNF formula (i.e., every clause has the form $(x \vee y)$ or $(\neg x \vee \neg y)$) can be solved in polynomial time.*

Before proving the theorem, we recall some graph-theoretic notions.

Definition 2 (Cut and Minimum Cut)**.** *Let $G = (V, E)$ be an undirected graph with non-negative edge weights.*

- *A* cut *is a partition of V into two disjoint sets S and $T = V \setminus S$.*
- *The* weight *of the cut (S, T) is the sum of the weights of all edges with one endpoint in S and the other in T.*

 doi:10.20944/preprints202409.2053.v20

- *A* minimum cut *(or* min-cut*) is a cut of minimum weight. Its value is denoted by* $\lambda(G)$.

The global minimum cut of a graph can be computed in polynomial time; for example, the deterministic algorithm of Stoer and Wagner [12,13] runs in

$$O\big(|V|\,(|E|+|V|)\,\log|V|\big).$$

Proof of Theorem 2. Let Φ be a monotone 2CNF formula with clause set

$$\mathcal{C} = \mathcal{C}^+ \cup \mathcal{C}^-,$$

where

- $\mathcal{C}^+$ consists of clauses of the form $(x \vee y)$,
- $\mathcal{C}^-$ consists of clauses of the form $(\neg x \vee \neg y)$.

Let V be the set of variables of Φ, and let $m = |\mathcal{C}|$ be the total number of clauses.

Graph Construction

Build an undirected graph $G = (V, E)$ as follows:

- For each clause $(x \vee y) \in \mathcal{C}^+$, add edge $\{x, y\}$ of weight 1.
- For each clause $(\neg x \vee \neg y) \in \mathcal{C}^-$, add edge $\{x, y\}$ of weight 1.

When parallel edges occur, they are merged into one edge whose weight equals their multiplicity. For instance, the presence of both $(x \vee y) \in \mathcal{C}^+$ and $(\neg x \vee \neg y) \in \mathcal{C}^-$ results in an edge $\{x, y\}$ with weight 2.

Assignment and Cut

Fix a truth assignment $\tau : V \to \{\text{true}, \text{false}\}$ and define the cut

$$S = \{v \in V : \tau(v) = \text{false}\}, \qquad T = \{v \in V : \tau(v) = \text{true}\}.$$

Unsatisfied Clauses

A clause is unsatisfied under τ exactly when both literals evaluate to false:

- $(x \vee y)$ is unsatisfied iff $x, y \in S$,
- $(\neg x \vee \neg y)$ is unsatisfied iff $x, y \in T$.

In both cases, the corresponding edge $\{x, y\}$ lies entirely within one side of the cut and does not cross (S, T).

Thus,

$$\#\{\text{unsatisfied clauses}\} = m - \text{weight}(S, T).$$

Optimization Equivalence

Maximizing the number of unsatisfied clauses is equivalent to minimizing the cut weight:

$$\max_{\tau} \#\{\text{unsatisfied clauses under } \tau\} = m - \lambda(G).$$

An optimal assignment is obtained from any minimum cut (S, T) by setting all variables in S to `false` and all variables in T to `true` (or vice versa). Since G has $O(m)$ edges and $\lambda(G)$ can be computed in polynomial time, the problem is solvable in polynomial time. □

We show that the problem MONOTONE-MIN-3SAT is solvable in polynomial time.

Theorem 3. *The problem* MONOTONE-MIN-3SAT *can be solved in polynomial time.*

 doi:10.20944/preprints202409.2053.v20

Proof. Consider a monotone 3CNF formula and a single positive clause

$$c_i = (x \vee y \vee z).$$

By Theorem 2, maximizing the number of unsatisfied clauses in a monotone 2CNF formula (clauses of the form $(x \vee y)$ or $(\neg x \vee \neg y)$) is solvable in polynomial time. We reduce MONOTONE-MIN-3SAT to this problem by replacing each clause c_i with a small monotone 2CNF gadget whose maximum number of unsatisfied clauses reflects whether c_i is satisfied.

Positive Clause Gadget

For $(x \vee y \vee z)$, introduce auxiliary variables x_i, y_i, z_i, w_i and a pivot variable w. Define

$$r(x, y, z, w) = \bigwedge_{j=1}^{10} C_j,$$

where the clauses are:

$$\begin{aligned}
&(x \vee x_i), \quad (y \vee y_i), \quad (z \vee z_i), \quad (\neg w \vee \neg w_i),\\
&(\neg x \vee \neg y), \quad (\neg y \vee \neg z), \quad (\neg z \vee \neg x),\\
&(x \vee w), \quad (y \vee w), \quad (z \vee w).
\end{aligned}$$

Fix $x_i = y_i = z_i =$ false; $w_i =$ true and analyze the minimum number of satisfiable clauses:

- **All x, y, z true:** $w =$ true yields 6 satisfied clauses; $w =$ false yields 7. Minimum: 6.
- **Exactly two true:** Both $w =$ true and $w =$ false yield 7. Minimum: 7.
- **Exactly one true:** $w =$ false yields 6; $w =$ true yields 7. Minimum: 6.
- **All false:** $w =$ false yields 4; $w =$ true yields 6. Minimum: 4.

Thus:

$$\#\text{maximum unsat} = \begin{cases} 4 & \text{if } (x \vee y \vee z) \text{ is satisfied,} \\ 6 & \text{if } (x \vee y \vee z) \text{ is unsatisfied.} \end{cases}$$

Negative Clause Gadget

For $(\neg x \vee \neg y \vee \neg z)$, introduce x_i, y_i, z_i, w_i, w and define:

$$\begin{aligned}
&(\neg x \vee \neg x_i), \quad (\neg y \vee \neg y_i), \quad (\neg z \vee \neg z_i), \quad (w \vee w_i),\\
&(x \vee y), \quad (y \vee z), \quad (z \vee x),\\
&(\neg x \vee \neg w), \quad (\neg y \vee \neg w), \quad (\neg z \vee \neg w).
\end{aligned}$$

A symmetric case analysis shows the same gap: 4 maximum unsatisfied clauses if the original clause is satisfied, 6 if unsatisfied.

Global Construction

Form a 2CNF formula R by replacing each clause of the original monotone 3CNF with its gadget. The size of R is linear in the original formula.

Each clause contributes either 4 or 6 maximum unsatisfied gadget clauses depending on satisfaction. Hence maximizing unsatisfied clauses in R is equivalent (up to an additive constant) to maximizing unsatisfied clauses in the original formula. Since the former is solvable in polynomial time, so is the latter.

Therefore, MONOTONE-MIN-3SAT is solvable in polynomial time. □

This is the main result.

 doi:10.20944/preprints202409.2053.v20

Theorem 4 (Main Theorem). *$P = NP$.*

Proof. This is a direct consequence of Theorems 1, 2 and 3. □

4. Conclusions

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 [3,4]. 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 [3,4].
- **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 [3,4]. The ability to efficiently analyze massive datasets would provide unparalleled insights in social sciences, economics, and healthcare, unlocking hidden patterns and correlations [3,4].
- **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 [3,4]. While the cryptographic landscape would face challenges, it would also present opportunities to develop new, provably secure encryption methods [3,4].
- **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 [3,4].

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: The author would like to thank Iris, Marilin, Sonia, Yoselin, and Arelis 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: December 4, 2025.
2. Sudan, M. The P vs. NP problem. http://people.csail.mit.edu/madhu/papers/2010/pnp.pdf, 2010. Accessed: December 4, 2025.
3. Fortnow, L. Fifty years of P vs. NP and the possibility of the impossible. *Communications of the ACM* **2022**, *65*, 76–85. https://doi.org/10.1145/3460351.
4. Aaronson, S. $P \stackrel{?}{=} NP$. *Open Problems in Mathematics* **2016**, pp. 1–122. https://doi.org/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. https://doi.org/10.1137/0204037.

 doi:10.20944/preprints202409.2053.v20

6. Razborov, A.A.; Rudich, S. Natural Proofs. *Journal of Computer and System Sciences* **1997**, *1*, 24–35. https://doi.org/10.1006/jcss.1997.1494.
7. Wigderson, A. *Mathematics and Computation: A Theory Revolutionizing Technology and Science*; Princeton University Press: Princeton, NJ, 2019.
8. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 2nd ed.; MIT Press: Cambridge, MA, 2001.
9. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman: San Francisco, CA, 1979.
10. Kohli, R.; Krishnamurti, R.; Mirchandani, P. The Minimum Satisfiability Problem. *SIAM Journal on Discrete Mathematics* **1994**, *7*, 275–283. https://doi.org/10.1137/S0895480191220836.
11. Darmann, A.; Döcker, J. On Simplified NP-Complete Variants of Monotone 3-Sat. *Discrete Applied Mathematics* **2021**, *292*, 45–58. https://doi.org/10.1016/j.dam.2020.12.010.
12. Stoer, M.; Wagner, F. A Simple Min-Cut Algorithm. *Journal of the ACM* **1997**, *44*, 585–591. https://doi.org/10.1145/263867.263872.
13. NetworkX Developers. NetworkX: stoer_wagner. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.connectivity.stoerwagner.stoer_wagner.html, 2025. Accessed: December 4, 2025.