Is P=NP:History and Frontier of NPC Problem

XIANZHE MENG, Huazhong University of Science and Technology, China

ACM Reference Format:
Xianzhe Meng. 2025. Is P=NP:History and Frontier of NPC Problem. *J. ACM* 1, 1, Article 1 (January 2025), 10 pages. https://doi.org/1037103.7103710

Abstract

This paper delves into the theoretical underpinnings and algorithmic strategies surrounding NP-complete (NPC) problems, with a particular emphasis on the *P vs NP* conundrum. Emerging from Cook's 1971 demonstration of SAT's NP-completeness and Karp's subsequent reduction framework, NPC problems epitomize the most challenging problems within the NP class. The paper elucidates key complexity classes (P, NP, NPC, NP-hard), expounds NP-completeness through quintessential problems like TSP and Vertex Cover, and examines proof strategies for both $P = NP$ (including algebraic methods) and $P \neq NP$. It also contemplates practical solutions such as approximation algorithms and parameterized tractability, alongside real-world applications. Ultimately, the study accentuates the ongoing theoretical challenges and the interdisciplinary ramifications of NPC research.

keywords:NP-complete problems;time complexity;algorithm history

1 Introduction

In the course of computational science development, NP-complete problems have always occupied a pivotal position. Back in 1971, it was Stephen Cook who, in his groundbreaking paper *The Complexity of Theorem Proving Procedures*[1], first introduced the concept of NP-completeness by proving SAT to be NP-complete. This achievement heralded an in-depth exploration of this unique class of problems. Subsequently, Richard Karp harnessed polynomial-time reductions to demonstrate that 21 classical combinatorial problems, such as TSP and VCP, also fall into the NP-complete category, thereby significantly broadening the scope of this theoretical domain. At the heart of NP-complete problems lies the contentious issue of *"P vs NP"*. Designated as one of the *Millennium Prize Problems* by the Clay Mathematics Institute with a reward of one million dollars for its solution[2], this question underscores the profound significance and elusiveness of NP-complete problems within the academic community.

Author's address: Xianzhe Meng, U202410203@hust.edu.cn, Huazhong University of Science and Technology, Wuhan, Hubei, China.

ACM 0004-5411/2025/1-ART1
https://doi.org/1037103.7103710

Over the past five decades, research on NP-complete problems has yielded fruitful results[3]. However, there still exists a paucity of systematic research that comprehensively reviews their historical development, key theoretical breakthroughs, and technological evolution.Our study endeavors to provide theoretical references and directional insights for future research.

2 Definition and Origination

2.1 SAT Problem

The Satisfiability Problem (SAT), a cornerstone in propositional logic, also stands as the first problem proven to be NP-complete in computational complexity theory. In propositional logic, formulas are constructed from variables (such as $x_1, x_2, \ldots, x_n$) and logical operators ($\wedge$ for conjunction, $\vee$ for disjunction, and $\neg$ for negation).

Given a propositional logic formula φ, the SAT problem seeks to determine whether there exists an assignment of truth values (True/False) to the variables such that the formula φ evaluates to True. If such an assignment exists, the formula φ is deemed satisfiable; otherwise, it is unsatisfiable.

In more rigorous mathematical terms, let $X = \{x_1, x_2, \ldots, x_n\}$ denote the set of variables. A literal is either a variable x_i or its negation $\neg x_i$. A clause is a disjunction (logical OR) of literals. For instance, $C = (x_1 \vee \neg x_2 \vee x_3)$ represents a clause. A propositional formula φ is typically expressed in CNF, which is a conjunction of clauses:

$$\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m = \bigwedge_{i=1}^{m} C_i, \quad \text{where} \quad C_i = \bigvee_{j=1}^{k_i} l_{ij} \tag{1}$$

where l_{ij} signifies a literal, and k_i denotes the number of literals in clause C_i. The objective of the SAT problem is to ascertain whether there exists a truth-value assignment $\tau : X \to \{\text{True}, \text{False}\}$ such that $\tau(\varphi) = \text{True}$.

In 1971, Stephen Cook established in his pioneering paper that the SAT problem is NP-complete [1]. This conclusion was reached through the following two steps:

(1) **SAT belongs to the class NP**: For any given truth-value assignment, we can verify whether this assignment satisfies the formula φ within polynomial time. It merely requires step-by-step computation of the truth value of φ in accordance with the rules of logical operators, with a time complexity of $O(|\varphi|)$, where $|\varphi|$ represents the length of the formula φ.
(2) **Any problem in NP can be reduced to SAT**: Cook demonstrated that for any problem L in NP, there exists a polynomial-time reduction f such that $x \in L$ if and only if $f(x)$ is satisfiable. This implies that should a polynomial-time algorithm be discovered for solving the SAT problem, then all NP problems could be resolved in polynomial time.

2.2 What is NP Problem?

In the realm of complexity theory, decision problems revolve around being solved or verified within polynomial time. The definitions of these problems are based on their time complexity[3].

P Problem: Decision problems that can be solved in polynomial time.

NP Problem: Decision problems for which a YES answer can be certified, and this certificate can be verified in polynomial time. However, it remains uncertain whether such problems can be solved in polynomial time.

NP-Complete(NPC) Problem: An NP problem to which all other NP problems can be reduced. Solving an NPC problem would yield solutions for all NP problems.

NP-Hard Problem: A problem H is NP-Hard if, for every $L \in NP$, there exists a polynomial-time reduction from L to H. Formally:

$$\forall L \in NP, \quad L \leq_p H \tag{2}$$

where $\leq_p$ represents a polynomial-time reduction.

The relationships among these definitions are illustrated in Image1 and Image2:

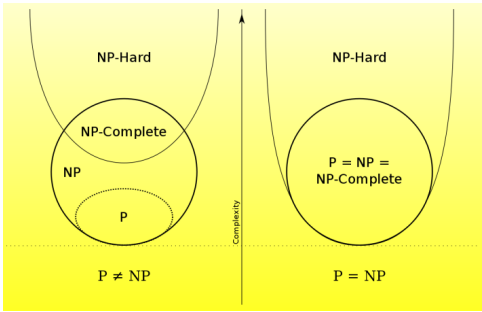

Image 1

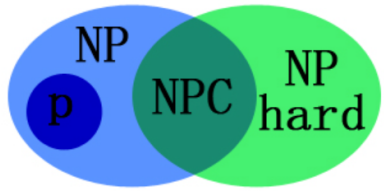

Image 2

2.3 History of Study

Table 1. Key Scientists and Study History of NP Problem[3]

Scientist	Year	Contribution	Significance
Manuel Blum	1967	Developed Blum complexity theory.	Established the foundational tools.
Stephen Cook	1971	Demonstrated the NP-completeness of SAT in "The Complexity of Theorem-Proving Procedures".	Established the concept of NPC.
Richard Karp	1972	Linked 21 problems to NP-completeness through reductions.	Expanded the scope of NP-hardness.

Continued on next page

Continued from previous page

Scientist	Year	Contribution	Significance
Leonid Levin	1973	Independently discovered NP-completeness.	Reinforced the theoretical foundation.
Mihir Bellare	1993	Contributed to the PCP theorem.	Connected NP to proof complexity.
Christos Papadimitriou	1994	Co-authored *Computational Complexity*.	Formalized complexity theory.
Johan Håstad	1997	Proved inapproximability results for MAX-3SAT.	Set limits on approximation algorithms.
Oded Goldreich	2001	Linked NP to cryptography in *Foundations of Cryptography*.	Enabled practical applications.
Dana Moshkovitz	2019	Advanced derandomization techniques.	Bridged the gap between P and NP.

3 Classical Examples

3.1 TSP

The Traveling Salesman Problem (TSP) stands as a classic combinatorial optimization problem. Given a set of n cities and the distances $d(i, j)$ between each pair of cities i and j, the objective is to identify the shortest possible route that visits each city exactly once and returns to the origin city. Mathematically, representing the route as a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of the cities, where π_1 denotes the starting city and $\pi_{n+1} = \pi_1$, the goal is to minimize the total distance:

$$\text{Minimize} \quad \sum_{i=1}^{n} d(\pi_i, \pi_{i+1})$$

The TSP manifests in two primary variants:

(1) **Symmetric TSP**: $d(i, j) = d(j, i)$ for all i and j, implying that the distance from city i to city j is identical to that from j to i.
(2) **Asymmetric TSP**: $d(i, j) \neq d(j, i)$ in general, often emerging in transportation networks with one-way streets or varying travel times in opposite directions.

3.1.1 Early Conceptions and Origins (1800s - 1930s) The concept of the TSP can be traced back to the 19th century.[4] In 1832, a German textbook on traveling salesmanship outlined a problem akin to the TSP, albeit not formalized as a mathematical problem. The first formal study of the TSP is attributed to the Irish mathematician William Rowan Hamilton and the British mathematician Thomas Kirkman in the 1850s. Hamilton's Icosian game, which involved finding a cycle that traverses each vertex of a dodecahedron exactly once, served as a precursor to the TSP. However, it focused on the existence of a cycle rather than optimizing a cost function.

The modern formulation of the TSP as an optimization problem emerged in the 1930s, primarily within the context of operations research and industrial applications. Mathematicians and researchers began to explore how to efficiently plan routes for salesmen,

delivery trucks, and other scenarios requiring traversal of multiple locations in an optimal manner.

3.1.2 Exact Algorithms: Early Approaches (1950s - 1960s) In the early stages of research, exact algorithms for the TSP centered on brute-force search and dynamic programming.

- **Brute - Force Search**: The most straightforward approach involves enumerating all possible permutations of the n cities and calculating the total distance of each tour. However, this method entails a time complexity of $\Theta(n!)$, rendering it impractical even for moderately large n. For example, when $n = 20$, there are $20! \approx 2.43 \times 10^{18}$ potential tours[4].
- **Held - Karp Algorithm (1962)**: Developed by Michael Held and Richard M. Karp, this dynamic programming algorithm addresses the TSP with a time complexity of $\Theta(n^2 2^n)$[5]. It operates by computing the shortest path to each subset of cities, effectively reducing the search space compared to brute-force methods. Despite its exponential nature, it marked a significant improvement for small-to-medium-sized instances and remains a benchmark for exact TSP solvers.

3.1.3 Computational Complexity and NP - Completeness (1970s) In 1972, Richard Karp established the NP - Hardness of the TSP in his seminal paper "Reducibility Among Combinatorial Problems"[6]. He demonstrated that the decision version of the TSP (i.e., given a distance bound k, does there exist a tour with a total distance of at most k) is NP - Complete. This implies that should a polynomial - time algorithm be found for the TSP, it would entail $P = NP$, thereby resolving one of the most significant open problems in computer science. This proof represented a major milestone, as it positioned the TSP as a fundamental problem in computational complexity studies.

3.1.4 Heuristic and Approximation Algorithms (1970s - Present) Given the computational expense of exact algorithms for large n, researchers have developed numerous heuristic and approximation algorithms to obtain near - optimal solutions within polynomial time [7].

- **Christofides' Algorithm (1976)**: One of the most renowned approximation algorithms for the metric TSP (where distances satisfy the triangle inequality, $d(i, j) + d(j, k) \geq d(i, k)$ for all i, j, k), Christofides' algorithm ensures a solution no more than 1.5 times the optimal tour length[8]. It integrates minimum spanning trees, matching, and path - merging techniques.
- **2 - Opt Heuristic (1950s)**: A simple yet effective local search heuristic. Starting with an initial tour, it iteratively enhances the tour by removing two edges and reconnecting the remaining paths in an alternative manner to reduce the total distance. Although it does not guarantee optimality, it can swiftly identify good solutions and is frequently employed as a subroutine in more complex algorithms.
- **Metaheuristics**: In recent decades, metaheuristics such as genetic algorithms, simulated annealing, and ant colony optimization have been extensively applied to the TSP. Drawing inspiration from natural phenomena (e.g., evolution, annealing in

physics, ant foraging behavior), these algorithms more effectively explore the solution space. For instance, genetic algorithms utilize operations like crossover and mutation on a population of candidate tours to evolve better solutions over time.

3.1.5 Modern Developments and Applications With the advancement of computing power and the emergence of new technologies, TSP research continues to evolve:

- **Parallel and Distributed Computing**: Researchers have investigated parallelizing TSP algorithms to leverage multi - core processors and distributed computing systems. This enables faster computation of solutions, particularly for large - scale instances.
- **Machine Learning and Data - Driven Approaches**: Recent research has explored employing machine learning techniques to predict good solutions for the TSP. For example, neural networks can be trained on a set of TSP instances to identify patterns and generate near - optimal tours more efficiently.

In conclusion, the Traveling Salesman Problem has served as a central topic in combinatorial optimization and computational complexity for over a century. Despite significant progress in algorithm development and complexity understanding, the pursuit of more efficient solutions and the quest to prove or disprove $P = NP$ continue to drive research in this field.

3.2 VCP

The Vertex Cover Problem (VCP) constitutes a fundamental combinatorial optimization problem in graph theory[9]. Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, a vertex cover is a subset $C \subseteq V$ such that every edge $e \in E$ is incident to at least one vertex in C. The objective is to identify the smallest possible vertex cover (minimum vertex cover). Formally:

Vertex Cover: A subset $C \subseteq V$ qualifies as a vertex cover if for every edge $(u, v) \in E$, either $u \in C$ or $v \in C$ (or both). The minimum vertex cover problem seeks to minimize $|C|$.

The decision version of VCP inquires:

> Given a graph G and an integer k, does there exist a vertex cover C with $|C| \leq k$?

3.2.1 Early Conceptions and Origins (1970s) The formal investigation of the Vertex Cover Problem emerged in the 1970s as part of the broader exploration into NP-completeness. The problem was among the first 21 problems proven to be NP-complete by Richard Karp in his landmark 1972 paper[6] . Although the concept of vertex covers dates back to earlier graph theory research, its classification as a canonical NP-complete problem solidified its importance in computational complexity theory.

3.2.2 Exact Algorithms: Early Approaches (1970s-1990s) Initial efforts to solve VCP centered on exact algorithms, primarily through brute-force enumeration and dynamic programming:

- **BFS**: This simplest approach involves checking all subsets of vertices of size k for k ranging from 1 to $|V|$. However, it entails a time complexity of $\Theta(2^n \cdot m)$, where $n = |V|$ and $m = |E|$, rendering it infeasible for large graphs.
- **Buss's Kernelization (1993)**: An early parameterized algorithm by Buss[10] demonstrated that VCP can be reduced to an equivalent instance (kernel) of size $\Theta(k^2)$ in polynomial time, leading to an $\Theta(2^k \cdot k^2 + n^2)$ algorithm for fixed k. This represented a significant improvement for small k.

3.2.3 Computational Complexity and NP-Completeness (1970s) Karp's 1972 reduction from 3-SAT to VCP established its NP-completeness. This proof demonstrated that any instance of 3-SAT (a known NP-complete problem) can be transformed into an equivalent instance of VCP in polynomial time[6] . This result had profound implications for approximation algorithms, as it suggested that VCP is unlikely to possess a polynomial-time exact solution unless $P = NP$.

3.2.4 Approximation Algorithms (1970s-Present) Owing to its NP-hardness, research shifted towards polynomial-time approximation algorithms:

- **2-Approximation (1970s)**: A simple greedy algorithm that repeatedly selects an edge and adds both its endpoints to the cover yields a 2-approximation[11]. This remains one of the most widely utilized algorithms for practical applications.
- **LP-Based Approximation (1980s)**: Utilizing linear programming relaxation, researchers developed a $\frac{2}{3}$-approximation algorithm for bipartite graphs.[12]
- **Inapproximability Results (2005)**: Dinur and Safra demonstrated that VCP cannot be approximated within a factor of 1.36 unless $P = NP$[13]. This result established a theoretical limit on approximation algorithms.

3.2.5 Parameterized Complexity (1990s-Present) The emergence of parameterized complexity theory led to significant advancements for VCP:

- **Fixed-Parameter Tractable (FPT) Algorithms**: In 1995, Chen et al. developed an $\Theta(1.286^k \cdot n)$ algorithm employing branching and kernelization techniques[14]. Subsequent improvements reduced the exponent to $\Theta(1.2738^k + kn)$ [15].
- **Kernelization Lower Bounds**: In 2014, Dell and van Melkebeek established that VCP does not admit a polynomial kernel unless $NP \subseteq coNP/poly$[16].

3.2.6 Modern Developments and Applications Recent research has concentrated on algorithmic engineering, parallel and distributed algorithms, machine learning, and heuristics. These advancements have deepened our understanding of VCP and facilitated numerous real-world applications, including:

(1) Sensor networks (minimizing active nodes while covering all edges).
(2) Fault detection in circuits (identifying critical components).
(3) Social network analysis (identifying influencers in viral marketing).

In summary, the Vertex Cover Problem has served as a cornerstone of computational complexity theory for decades. Its study has driven innovations in approximation algorithms, parameterized complexity, and practical optimization techniques. While exact solutions remain intractable for large graphs, ongoing research continues to expand the boundaries of computational feasibility.

4 Algorithmic Strategies

4.1 Proof Strategies for $P = NP$

Establishing $P = NP$ necessitates either constructing a polynomial-time algorithm for an NP-complete problem $\Pi \in$ NPC or demonstrating that non-deterministic Turing machines (NTMs) can be simulated deterministically within polynomial time. Below are the principal approaches:

4.1.1 Constructive Algorithms for NP-Complete Problems

The most straightforward method involves designing a polynomial-time algorithm for an NP-complete problem, such as 3-SAT or TSP.

Algebraic Methods Translating NP-complete problems into algebraic systems represents one approach. For example, encoding a 3-SAT formula ϕ as a system of polynomial equations over the finite field F_2:

$$\phi(x_1, \ldots, x_n) \equiv \bigwedge_{i=1}^{m} C_i(x_1, \ldots, x_n) \tag{3}$$

Each clause C_i corresponds to a polynomial constraint. Should such systems be solvable in $\Theta(p(n))$ time for some polynomial p, it would imply $P = NP$. However, general polynomial system solving is NP-hard, and no efficient algorithm is currently known.

Quantum Computing Quantum algorithms leverage superposition and entanglement to explore solution spaces. For instance, Shor's algorithm factors integers in $\Theta((\log N)^3)$ time, exponentially faster than classical algorithms. Nevertheless, no quantum algorithm has yet solved an NP-complete problem in polynomial time. Quantum annealing shows promise for optimization but lacks theoretical guarantees for $P = NP$.

Structural Insights Exploiting hidden structures within NP-complete problems presents another avenue. For example, planar graphs admit polynomial-time solutions for certain NP-complete problems, such as 4-colorability (decidable in $\Theta(n^2)$ time) and TSP (approximable within a factor of $1+\varepsilon$ in polynomial time). However, these results do not generalize to arbitrary graphs.

4.1.2 Logical and Meta-Complexity Approaches

Simulating Non-Determinism Proving that the exponential branching of NTMs can be compressed into polynomial time stands as a key direction. Approaches include derandomization, where the hypothesis NP $\subseteq$ BPP combined with robust pseudorandom generator (PRG) assumptions (e.g., E requiring circuits of size $2^{\Omega(n)}$) would imply NP $\subseteq$ P.

Alternatively, combinatorial bounds on the number of accepting paths in an NTM, utilizing techniques like expander graphs, could facilitate polynomial-time simulations.

Logical Characterizations Relating P and NP to logical systems offers another perspective. For example, Fagin's Theorem posits that NP comprises the set of languages expressible in existential second-order logic. A proof demonstrating that P and NP coincide within a logical framework (e.g., fixed-point logic with counting) would resolve the P vs NP problem.

4.2 Proof Strategies for $P \neq NP$

Demonstrating $P \neq NP$ requires showing that some $\Pi \in NP$ demands super-polynomial time. Key approaches encompass:

Diagonalization and Oracle Separations Classical diagonalization, which constructs a language $L \in NP$ that diagonalizes against all polynomial-time DTMs, fails due to the relativization barrier: oracles A and B exist such that $P^A = NP^A$ and $P^B \neq NP^B$. This indicates that any proof of $P \neq NP$ must employ non-relativizing techniques. While interactive proof systems, such as MIP* = RE, provide insights into complexity class separations, they are too powerful to directly imply $P \neq NP$.

Circuit Complexity Establishing super-polynomial lower bounds on the size of Boolean circuits required to compute NP-complete functions represents a central approach. Razborov's 1985 result showed that monotone circuits for CLIQUE require size $2^{\Omega(n^{1/4})}$, but this does not extend to general circuits. The natural proofs barrier posits that any "natural" proof of a circuit lower bound would violate cryptographic assumptions. Similarly, the algebrization barrier[17]demonstrates that techniques employing linear algebra or arithmetization cannot separate P and NP.

Proof Complexity Associating the difficulty of solving NP problems with the length of their proofs in formal systems presents another avenue. For example, proving that tautologies encoding NP-hard problems (e.g., TSP) require super-polynomial-length proofs in extended Frege systems would imply $NP \neq coNP$, a result stronger than $P \neq NP$. Craig's interpolation theorem links proof complexity and circuit complexity, yet no such lower bounds are known for NP-complete problems.

5 Conclusion and Further Thinking

Since the formalization of NP-completeness over four decades ago, these problems have remained central to theoretical computer science. Foundational work by Cook in 1971 and Karp in 1972 revealed that a wide array of combinatorial decision problems (e.g., SAT, TSP, Vertex Cover) are all inter-reducible, forming the class of the "hardest problems in NP" [6]. As Garey and Johnson emphasized, the question of whether these NP-complete problems are inherently intractable stands as "one of the foremost open questions of contemporary mathematics and computer science" [11]. Thus, the P versus NP problem has become a touchstone for understanding the limits of efficient computation.

Over the years, a vast body of research has developed around NP-complete problems.Despite this extensive research, the fundamental P vs NP question remains unresolved. But we have research on how NPC problem can be solved,which is summarized in this paper.

Beyond theory, NP-complete problems have profound practical relevance. Classic examples like the Boolean satisfiability problem, the traveling salesman problem, and the Vertex Cover problem emerge in diverse applications. For instance, the Steiner tree problem (an NP-hard variant closely related to vertex cover) has direct applications in circuit design and network topology[18]. Therefore,research on NP-complete problems remains a fertile frontier.Study on NPC problem should be emphasized,which remains important to human society.

References

[1] Cook and A. Stephen. 1971. The complexity of theorem-proving procedures. *ACM* (1971), 151–158.

[2] J Carlson, A Jaffe, and A Wiles. 2006. *The Millennium Prize Problems.* The millennium prize problems.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. [n. d.]. Introduction to Algorithms, 3rd Ed. ([n. d.]).

[4] Gregory Gutin, Abraham Punnen, Alexander Barvinok, Edward Kh. Gimadi, and Anatoliy I. Serdyukov. 2001. The traveling salesman problem and its variations. *Springer US* (2001).

[5] Held Richard M. Karp. 1962. A Dynamic Programming Approach to Sequencing Problems. *Journal of the Society for Industrial & Applied Mathematics* 10, 1 (1962), 196–210.

[6] R. Karp. 1972. Reducibility among combinatorial problems" in complexity of computer computations. (1972).

[7] Sriyani Violina. 2021. Analysis of Brute Force and Branch & Bound Algorithms to solve the Traveling Salesperson Problem (TSP). (2021).

[8] Daniel J Rosenkrantz, Richard E Stearns, and Philip M. Lewis Ii. 1976. Worst-case analysis of a new heuristic for the traveling salesman problem. (1976).

[9] Wenjun Li, Yang Ding, Yongjie Yang, R. Simon Sherratt, Jong Hyuk Park, and Jin Wang. 2020. Parameterized algorithms of fundamental NP-hard problems: a survey. *SpringerOpen* 1 (2020).

[10] Josep Díaz, Jordi Petit, and Dimitrios M. Thilikos. 2006. Kernels for the Vertex Cover Problem on the Preferred Attachment Model. *Springer-Verlag* (2006).

[11] Michael R Garey and David S Johnson. 1983. Computers and Intractability: A Guide to the Theory of NP-Completeness. *W. H. Freeman* (1983).

[12] Martin W. P. Savelsbergh, R. N. Uma, and Joel Wein. 1998. An experimental study of LP-based approximation algorithms for scheduling problems. *Informs Journal on Computing* 17, 1 (1998).

[13] I. Dinur and S. Safra. 2005. Analytical Approach to PCPs: The Characterization of NP. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. 2–12.

[14] Jianer Chen, Iyad A. Kanj, and Weijia Jia. 2001. Vertex Cover: Further Observations and Further Improvements. *Journal of Algorithms* 41, 2 (2001), 280–301.

[15] Fedor V. Fomin and Petteri Kaski. 2013. Exact Exponential Algorithms. *Communications of the Acm* 56, 3 (2013), 80–88.

[16] Holger Dell and Dieter Van Melkebeek. 2014. Satisfiability Allows No Nontrivial Sparsification Unless The Polynomial-Time Hierarchy Collapses. *Journal of the ACM (JACM)* 61, 4 (2014), 1–27.

[17] Scott, Aaronson, Avi, and Wigderson. 2009. Algebrization: A New Barrier in Complexity Theory. *ACM Transactions on Computation Theory* 1, 1 (2009), 1–54.

[18] Michael Sipser. 1973. *Introduction to the theory of computation.* Introduction to the theory of computation.