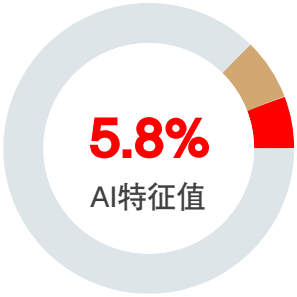
AIGC检测 · 全文报告单

NO:CNKIAIGC2025FG_202506106309094　　检测时间：2025-06-06 21:16:00

篇名： NPC problem
作者： 孟宪喆
单位： 华中科技大学
文件名: NPC problem.pdf

全文检测结果　知网AIGC检测　https://cx.cnki.net



AI特征值：5.8%
AI特征字符数：1581
总字符数：27060

- AI特征显著（计入AI特征字符数）
- AI特征疑似（未计入AI特征字符数）
- 未标识部分

AIGC片段分布图

前部20%	中部60%	后部20%
AI特征值：19.5%	AI特征值：3.3%	AI特征值：0.0%
AI特征字符数：1053	AI特征字符数：528	AI特征字符数：0

分段检测结果

序号	AI特征值	AI特征字符数 / 章节(部分)字符数	章节(部分)名称
1	6.3%	1053 / 16716	NPC problem_第1部分
2	5.1%	528 / 10344	NPC problem_第2部分

1. NPC problem_第1部分

AI特征值：6.3%　　AI特征字符数 / 章节(部分)字符数：1053 / 16716

片段指标列表

序号	片段名称	字符数	AI特征	
1	片段1	964	疑似	5.8%
2	片段2	1053	显著	6.3%

原文内容

Is P=NP:History and Frontier of NPC Problem

Xianzhe Meng,U202410203,HUST 2025-06-06

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 prob- lems like TSP and Vertex Cover, and examines proof strategies for both P = N P (including algebraic methods) and P /= N P . 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.

5.8%(964)

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 (Cook, 1971), 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

6.3%(1053)

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 (Clay Mathematics Institute, 2000), this question underscores the profound significance and elusiveness of NP-complete problems within the academic community.

Over the past five decades, research on NP-complete problems has yielded fruitful results (Cormen et al., 2009). However, there still exists a paucity of systematic research that compre- hensively reviews their historical development, key theoretical breakthroughs, and technological evolution. This paper endeavors to

2Definition and Origination

2.1SAT 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 x1, x2, . . . , xn) and logical operators (∧ for conjunction, ∨ for disjunction, and ¬ 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 = {x1, x2, . . . , xn} denote the set of variables. A literal is either a variable xi or its negation ¬xi. A clause is a disjunction (logical OR) of literals. For instance, C = (x1 ∨ ¬x2 ∨ x3) represents a clause. A propositional formula Φ is typically expressed in CNF, which is a conjunction of clauses:

$\wedge_{i=1}^{m} \vee_{j=1}^{k_i} \Phi = C1 \wedge C2 \wedge \cdots \wedge Cm = Ci$, where $Ci = l_{ij}$

Here, lij signifies a literal, and ki denotes the number of literals in clause Ci. The objective of the SAT problem is to ascertain whether there exists a truth-value assignment $\tau : X \rightarrow \{True, False\}$ such that $\tau(\Phi) = True$.

In 1971, Stephen Cook established in his pioneering paper that the SAT problem is NP- complete (Cook, 1971). 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 $O(|\Phi|)$, where $|\Phi|$ represents the length of the formula

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 (Cook, 1971). 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.

2.2What 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 (Cormen et al., 2009).

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 certifi- cate 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:

$\forall L \in NP, L \leqslant_p H$

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

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

Image1 Image2

2.3History of Study

Table 1: Key Scientists and Study History of NP Problem

Scientist YearContribution

Significance

Manuel 1967Developed Blum complexity the-Established the founda-
Blum
ory.
tional tools.
Stephen 1971Demonstrated the NP-Established the concept of
Cook
completeness of SAT in ”TheNPC.
Complexity of Theorem-Proving
Procedures”.
Richard
1972Linked 21 problems to NP-Expanded the scope of NP-
Karp
completeness through reductions.hardness.
Leonid
1973Independently discovered NP-Reinforced the theoretical
Levin
completeness.
foundation.
Mihir Bel-1993Contributed to the PCP theorem.Connected NP to proof
lare
complexity.
Christos1994Co-authored ComputationalFormalized complexity the-
Papadim-
Complexity.
ory.
itriou
Johan
1997Proved inapproximability resultsSet limits on approximation
Håstad
for MAX-3SAT.
algorithms.
Oded Gol-2001Linked NP to cryptography inEnabled practical applica-
dreich
Foundations of Cryptography.
tions.
Dana

2019Advanced derandomization tech-Bridged the gap between P Moshkovitz

niques.

and NP.

3Classical Examples

3.1TSP

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 π = (π1, π2, . . . , πn) of the cities, where π1 denotes the starting city and πn+1 = π1, the goal is to minimize the total distance:

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

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) ≠d(j, i) in general, often emerging in transportation networks with one-way streets or varying travel times in opposite directions.

3.1.1Early Conceptions and Origins (1800s - 1930s)

The concept of the TSP can be traced back to the 19th century. In 1832, a German text- book 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 mathemati- cian 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 dodeca- hedron 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 eiciently plan routes for salesmen, delivery trucks, and other scenarios requiring traversal of multiple locations in an optimal manner.

3.1.2Exact 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. How- ever, 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 $\times$ 1018 potential tours.

• Held - Karp Algorithm (1962): Developed by Michael Held and Richard M. Karp, this dynamic programming algorithm addresses the TSP with a time complexity of $\Theta(n22n)$

(Held and Karp, 1962). 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.

3.1.3Computational Complexity and NP - Completeness (1970s)

In 1972, Richard Karp established the NP - Hardness of the TSP in his seminal paper ”Reducibility Among Combinatorial Problems” (Karp, 1972). 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 = N P , 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.

3.1.4Heuristic 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 (Yilmaz and Kaya, 2020).

• Christofides’ Algorithm (1976): One of the most renowned approximation algorithms for the metric TSP (where distances satisfy the triangle inequality, $d(i, j)+d(j, k) \geqslant d(i, k)$ for all i, j, k), Christofides’ algorithm ensures a solution no more than 1.5 times the

optimal tour length (Christofides, 1976). 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 eiciently.

• Real - World Applications: The TSP finds extensive practical applications, including logistics and transportation (Nguyen Thach et al., 2025) (e.g., delivery route planning), circuit board manufacturing (minimizing drill paths), DNA sequencing (ordering DNA fragments), and robotics (path planning for mobile robots).

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 eicient solutions and the quest to prove or disprove P = N P continue to drive research in this field.

3.2 VCP

The Vertex Cover Problem (VCP) constitutes a fundamental combinatorial optimization problem in graph theory. Given an undirected graph G = (V, E) with vertex set V and edge set E, a vertex cover is a subset C ⊆ V such that every edge e ∈ 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 ⊆ V qualifies as a vertex cover if for every edge (u, v) ∈ E, either u ∈ C or v ∈ 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| ⩽ 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 (Karp, 1972). 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.

3.2.2Exact Algorithms: Early Approaches (1970s-1990s)

Initial efforts to solve VCP centered on exact algorithms, primarily through brute-force enumeration and dynamic programming:

• Brute-Force Search: 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 Θ(2n • m), where n = |V | and m = |E|, rendering it infeasible for large graphs.

• Buss’s Kernelization (1993): An early parameterized algorithm by Buss demonstrated that VCP can be reduced to an equivalent instance (kernel) of size Θ(k2) in polynomial time, leading to an Θ(2k • k2 + n2) algorithm for fixed k (Buss, 1993). This represented a significant improvement for small k.

3.2.3Computational 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 (Karp, 1972). This result had profound implications for approximation algorithms, as it suggested that VCP is unlikely to possess a polynomial-time exact solution unless P = N P .

3.2.4Approximation Algorithms (1970s-Present)

Owing to its NP-hardness, research shifted towards polynomial-time approximation algo- rithms:

• 2-Approximation (1970s): A simple greedy algorithm that repeatedly selects an edge and adds both its endpoints to the cover yields a 2-approximation (Garey and Johnson, 1979). This remains one of the most widely utilized algorithms for practical applications.

• LP-Based Approximation (1980s): Utilizing linear programming relaxation, researchers developed a 23 -approximation algorithm for bipartite graphs.

• Inapproximability Results (2005): Dinur and Safra demonstrated that VCP cannot be approximated within a factor of 1.36 unless P = N P (Dinur and Safra, 2005). This result established a theoretical limit on approximation algorithms.

3.2.5Parameterized 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 $\Theta(1.286k \cdot n)$ algorithm employing branching and kernelization techniques (Chen et al., 2006). Subsequent improvements reduced the exponent to $\Theta(1.2738k + kn)$ (Fomin and Kratsch, 2016).

2. NPC problem_第2部分

AI特征值：5.1%　　AI特征字符数 / 章节(部分)字符数：528 / 10344

片段指标列表

序号	片段名称	字符数	AI特征	
3	片段1	528	显著	5.1%

4	片段2	891	疑似		8.6%

原文内容

• Kernelization Lower Bounds: In 2010, Dell and van Melkebeek established that VCP does not admit a polynomial kernel unless N P ⊆ coN P /poly (Dell and van Melkebeek, 2010).

3.2.6Modern Developments and Applications

Recent research has concentrated on algorithmic engineering, parallel and distributed algo- rithms, 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 com- plexity theory for decades. Its study has driven innovations in approximation algorithms, pa- rameterized complexity, and practical optimization techniques. While exact solutions remain intractable for large graphs, ongoing research continues to expand the boundaries of computa- tional feasibility.

5.1%(528)

4 Algorithmic Strategies

4.1 Proof Strategies for P = N P

Establishing P = N P necessitates either constructing a polynomial-time algorithm for an NP-complete problem Π ∈ 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 F2:

$\bigwedge_{i=1}^{m} \phi(x1, \ldots, xn) \equiv Ci(x1, \ldots, xn)$ (1)

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

Quantum Computing Quantum algorithms leverage superposition and entanglement to ex- plore solution spaces. For instance, Shor's algorithm factors integers in $\Theta((\log N)3)$ time, expo- nentially 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 = N P .

Structural Insights Exploiting hidden structures within NP-complete problems presents an- other avenue. For example, planar graphs admit polynomial-time solutions for certain NP- complete problems, such as 4-colorability (decidable in $\Theta(n2)$ 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-DeterminismProving that the exponential branching of NTMs can be compressed into polynomial time stands as a key direction. Approaches include derandomiza- tion, 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, combi- natorial 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 N P to logical systems offers another perspective. For example, Fagin's Theorem posits that N P comprises the set of languages expressible in existential second-order logic. A proof demonstrating that P and N P coincide within a logical framework (e.g., fixed-point logic with counting) would resolve the P vs N P problem.

4.2 Proof Strategies for P $\neq$NP

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

4.2.1Diagonalization and Oracle Separations

Classical diagonalization, which constructs a language L ∈ N P that diagonalizes against all polynomial-time DTMs, fails due to the relativization barrier: oracles A and B exist such that P A = N P A and P B/= N P B. This indicates that any proof of P/=N P 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 ≠NP.

4.2.2Circuit 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 2Ω(n1/ 4), but this does not extend to general circuits. The natural proofs barrier (Razborov and Rudich, 1997) posits that any "natural" proof of a circuit lower bound would violate cryptographic assumptions. Similarly, the algebrization barrier (Aaronson and Wigderson, 2009) demonstrates that techniques employing linear algebra or arithmetization cannot separate P and N P .

4.2.3 Proof Complexity

Associating the diiculty 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 N P /= coN P , a result stronger than P ≠N P . 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" (Karp, 1972). 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 mathe- matics and computer science" (Garey and Johnson, 1979). Thus, the P versus NP problem has become a touchstone for understanding the limits of eicient

computation.

Over the years, a vast body of research has developed around NP-complete problems. On the algorithmic front, researchers have devised sophisticated exact and heuristic methods: for instance, specialized exponential-time algorithms with clever pruning, as well as approximation and metaheuristic algorithms that yield near-optimal solutions for large instances (Juan et al., 2023). Many NP-hard optimization tasks in logistics and scheduling (such as vehicle routing and orienteering) are now addressed with such advanced heuristics under uncertainty (Juan et al., 2023). Meanwhile, theoretical work has deepened our understanding of NP-completeness through tools like probabilistically checkable proofs, parameterized complexity, and hardness- of-approximation frameworks. Investigators have also explored new computational paradigms —for example, quantum algorithms and circuit complexity models—seeking any breakthrough. Notably, quantum computing (e.g., Shor’s algorithm for factoring) has dramatically advanced some problems but has not yet produced polynomial-time solutions for general NP-complete problems.

Despite this extensive research, the fundamental P vs NP question remains unresolved. To date, no one has succeeded in proving that NP problems require super-polynomial time (which would imply P ≠ N P), nor in finding a general polynomial-time algorithm for any NP-complete problem (which would imply P = N P) (Garey and Johnson, 1979). Significant theoretical milestones (such as the PCP theorem and relativization results) have illuminated the structure of NP, but key challenges persist. In particular, known barriers in circuit-complexity and proof complexity (e.g., ”natural proof” limitations) suggest that truly novel ideas may be needed to settle the question. This enduring diiculty is underscored by the fact that even after a million- dollar Clay prize was offered for P vs NP, the problem remains open (Garey and Johnson, 1979).

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 (Sipser, 2012). Similarly, many

operations-research problems (integer programming, TSP routing, scheduling, etc.) are NP-complete in general; making them eiciently solvable would have ”enormous implications for logistics” and optimization in industry (Juan et al., 2023). Conversely, the presumed hardness of NP-complete problems underpins the security of modern cryptography. As noted in standard complexity literature, a constructive polynomial- time solution to an NP-complete problem (e.g., 3-SAT) would effectively ”break most existing cryptosystems,” compromising public-key and symmetric encryption (Juan et al., 2023). In practice, cryptographic protocols rely on the assumption that P/=N P , so any progress on the P vs NP question has immediate implications for cybersecurity.

Looking ahead, research on NP-complete problems remains a fertile frontier. Efforts persist on multiple fronts: enhancing approximation and fixed-parameter algorithms for challenging cases; designing superior heuristics and hybrid methods for practical instances; and exploring new theoretical techniques (stronger circuit lower bounds, algebraic methods, or perhaps entirely new models of computation). Even if P ≠ N P ultimately remains unproven, advances in related areas (such as complexity theory, proof systems, and optimization theory) continue to refine our understanding and often yield practical algorithmic improvements. In summary, the NP- complete class encapsulates a rich history and enduring challenge: the depth and breadth of work to date testify to its importance, and resolving its remaining questions would have profound consequences both in theory and in applications.

8.6%(891)

说明:

1、支持中、英文内容检测;
2、AI特征值=AI特征字符数/总字符数;
3、红色代表AI特征显著部分，计入AI特征字符数;
4、棕色代表AI特征疑似部分，未计入AI特征字符数;
5、检测结果仅供参考，最终判定是否存在学术不端行为时，需结合人工复核、机构审查以及具体学术政策的综合应用进行审慎判断。

https://cx.cnki.net
知网个人AIGC检测服务