Tunnell's Theorem and #P-Completeness

Frank Vega *

Posted Date: 20 November 2025

doi: 10.20944/preprints202511.1586.v1

Keywords: computational complexity; #P-completeness; Tunnell's theorem; Birch–Swinnerton-Dyer conjecture; congruent numbers; P versus NP

 doi:10.20944/preprints202511.1586.v1

Article

Tunnell's Theorem and #P-Completeness

Frank Vega

Information Physics Institute, 840 W 67th St, Hialeah, FL 33012, USA; vega.frank@gmail.com

Abstract

We propose a new hypothetical framework to explore the relationship between the Birch–Swinnerton-Dyer conjecture (BSD) and computational complexity theory. This paper introduces two central conjectures: a Reduction Conjecture, which posits the existence of a polynomial-time parsimonious reduction from #P-complete problems to the counting of integer solutions for Tunnell's equations, and a Solution Density Conjecture, which posits that these solution counts are sufficiently well-distributed. We then formally prove that if these two conjectures are true, a surprising conditional result follows: the assumptions of P = NP and the truth of the BSD conjecture would imply the collapse of the counting complexity class #P to FP. Our main conditional result is that if BSD is true, our conjectures hold, and the widely believed separation #P $\neq$ FP holds, then P $\neq$ NP. This work reframes the P versus NP problem as a question about two deep, open problems in number theory: the existence of a "complexity-to-arithmetic" reduction and the statistical distribution of solution counts to Tunnell's specific quadratic forms.

Keywords: computational complexity; #P-completeness; Tunnell's theorem; Birch–Swinnerton-Dyer conjecture; congruent numbers; P versus NP

MSC: 11G05; 68Q15; 11D09; 68Q17

---

1. Introduction

The P versus NP problem stands as one of the most fundamental open questions in computer science and mathematics [1,2]. This paper proposes a novel, albeit conjectural, pathway to connect the P versus NP question to the Birch–Swinnerton-Dyer conjecture (BSD) [3] through the lens of Tunnell's theorem [4].

1.1. Background on Complexity Classes

The complexity class #P, introduced by Valiant [5], consists of counting problems associated with NP decision problems. Formally, a function $f : \{0,1\}^* \to \mathbb{N}$ belongs to #P if there exists a polynomial-time verifier V and a polynomial p such that

$$f(x) = |\{y \in \{0,1\}^{p(|x|)} : V(x,y) = 1\}|.$$

A problem is #P-complete if every problem in #P reduces to it via a polynomial-time parsimonious reduction (or more generally, via polynomial-time reductions that preserve solution counts up to polynomial factors). Classic #P-complete problems include #SAT (counting satisfying assignments) and computing the permanent of a matrix [5].

The functional class FP consists of functions computable in polynomial time. It is widely conjectured that #P $\not\subseteq$ FP, as this would imply P $\neq$ NP and represent an even stronger separation [6].

 doi:10.20944/preprints202511.1586.v1

1.2. Background on Congruent Numbers and Tunnell's Theorem

A positive integer n is called a *congruent number* if it is the area of a right triangle with rational side lengths [7]. Equivalently, n is congruent if and only if the elliptic curve

$$E_n : y^2 = x^3 - n^2 x$$

has positive rank. The congruent number problem—determining whether a given n is congruent—is one of the oldest unsolved problems in number theory.

Tunnell [4] made remarkable progress by establishing a computationally tractable criterion. For a square-free integer n, define

$$C_n = \#\{(x, y, z) \in \mathbb{Z}^3 : n = 8x^2 + 2y^2 + 64z^2\},$$
$$D_n = \#\{(x, y, z) \in \mathbb{Z}^3 : n = 8x^2 + 2y^2 + 16z^2\}.$$

Theorem 1 (Tunnell [4]). *Let n be a square-free positive integer.*

1. *If n is even and congruent, then $2C_n = D_n$.*
2. *Conversely, if the Birch–Swinnerton-Dyer conjecture holds for the elliptic curve E_n, then $2C_n = D_n$ implies n is congruent.*

The *congruum* construction, dating back to Fibonacci, provides an explicit family of congruent numbers [8]. A congruum is an integer of the form

$$g = 4mn(m^2 - n^2)$$

for distinct positive integers $m > n$. Every congruum is the common difference in a Pythagorean progression, and multiplying a congruum by a perfect square yields another congruum.

1.3. Main Contributions and Framework

This paper does not claim a new proven result. Instead, it proposes a *framework* for attacking the P vs. NP problem by linking it to number theory. The key contributions are:

1. **The Solution Density Conjecture (Conjecture 3):** We conjecture that the solution counts D_n for congruent numbers are sufficiently dense and well-distributed, enabling a search algorithm (under P=NP) to find instances with polynomially-bounded counts.
2. **The Reduction Conjecture (Conjecture 5):** We conjecture the existence of a polynomial-time reduction R that maps an arbitrary #P-complete problem instance I to an even square-free congruent number n such that the solution count $\#I$ is polynomially related to D_n.
3. **A Conditional Algorithm:** We present an algorithm (Algorithm 1) and prove that *if* our conjectures hold and P=NP, it solves #P-complete problems in polynomial time.
4. **A Conditional Implication (Theorem 7):** We prove that if our conjectures are true, then the statement '(P = NP $\wedge$ BSD) $\implies$ (#P = FP)' holds. This provides a novel, albeit conjectural, pathway to relating these fundamental open problems.

1.4. Organization

Section 2 establishes notation and preliminary results. Section 3 develops ancillary results on congruum density, polynomial balance, and counting under complexity assumptions, and introduces the first of our key conjectures (Conjecture 3). Section 4 presents the main reduction and algorithm, built upon the central Reduction Conjecture (Conjecture 5). Section 5 discusses implications and the nature of these conjectures. Section 8 concludes with open questions.

 doi:10.20944/preprints202511.1586.v1

2. Preliminaries

2.1. Notation and Conventions

Throughout this paper, we use standard complexity-theoretic notation. For a function $f : \mathbb{N} \to \mathbb{N}$, we write $f(n) = \text{poly}(n)$ if there exist constants $c, k > 0$ such that $f(n) \leq c \cdot n^k$ for all sufficiently large n.

For an integer m, we denote by $|m|$ its bit length, i.e., $|m| = \lceil \log_2(\text{abs}(m) + 1) \rceil$, where $\text{abs}(\ldots)$ is the absolute value. For a computational problem instance I, we write $|I|$ for the size of its encoding.

2.2. Assumptions

We work under the following two critical assumptions throughout:

1. **P = NP:** The class of problems solvable in polynomial time equals the class of problems verifiable in polynomial time. This immediately implies NP = coNP = P.
2. **BSD:** The Birch–Swinnerton-Dyer conjecture holds for elliptic curves of the form $E_n : y^2 = x^3 - n^2 x$.

Under P = NP, several important consequences follow:

- Factorization $\in$ P, so extracting square-free parts is polynomial-time.
- Satisfiability testing $\in$ P.
- Existence and counting can be decided via binary search using NP and coNP oracles, both in P.

2.3. Diophantine Representations

A fundamental result connecting logic and number theory is Matiyasevich's theorem [9]:

Theorem 2 (Matiyasevich). *Every recursively enumerable set has a Diophantine representation. Specifically, for any NP problem, there exists a polynomial P with integer coefficients such that membership in the set is equivalent to solvability of $P(x_1, \ldots, x_k) = 0$ in integers.*

3. Ancillary Results and the First Conjecture

This section develops the technical machinery required for our hypothetical framework.

3.1. Congruum Density and Distribution

Lemma 1. *The number of distinct congruums up to X is $\Omega(X^{1/2})$.*

Proof. Each pair (m, n) with $1 \leq n < m \leq \sqrt{X}$ generates a distinct congruum

$$g = 4mn(m^2 - n^2) \leq 4m \cdot m \cdot m^2 = 4m^4 \leq X.$$

The number of such pairs is

$$\sum_{m=2}^{\lfloor \sqrt{X} \rfloor} (m - 1) = \frac{\lfloor \sqrt{X} \rfloor (\lfloor \sqrt{X} \rfloor - 1)}{2} = \Theta(X^{1/2}).$$

Distinctness follows from unique factorization and the specific form of congruums. □

Conjecture 3 (Solution Density Conjecture). *For any target count $T \in [1, 2^{\text{poly}(N)}]$ and size bound $B = \text{poly}(N)$, there exists an even square-free congruent number n with $|n| \leq B$ such that $D_n \in [T/B^2, T \cdot B^2]$.*

Moreover, under the assumptions P = NP and BSD, such an n can be found *in time* $\text{poly}(N)$.

Remark 1 (Justification and Difficulty). *The first part of this conjecture is a deep statement about the distribution of coefficients of modular forms. Lemma 1 shows there is a large supply of congruent numbers (at least $\Omega(2^{B/2})$ with bit length $O(B)$). The core of this conjecture is that the corresponding set of solution counts*

 doi:10.20944/preprints202511.1586.v1

$\{D_n\}$ is not pathologically sparse but "fills" the possible range densely enough for a search algorithm to succeed. Proving this would require significant new results in analytic number theory.

The second part, finding n, relies on P = NP. It assumes we can iterate through candidates, compute D_n for each (using Lemma 2), and find one in the target range, all in polynomial time. The existence is the hard number-theoretic part; the search is the complexity-theoretic part.

3.2. Counting via Binary Search

A crucial technique under P = NP = coNP is determining exact counts via binary search.

Lemma 2. *Under P = NP = coNP, for any NP problem instance I, the exact number of solutions can be computed in polynomial time.*

Proof. Let $V(I, w)$ be a polynomial-time verifier for I, where solutions w have length $p(|I|)$ for some polynomial p.

Define the decision problem: "Does I have at least k solutions?" This is in NP (guess k distinct solutions and verify). Its complement is in coNP. Under P = NP = coNP, both are in P.

Binary search over $k \in [0, 2^{p(|I|)}]$ determines the exact count in $O(p(|I|))$ oracle calls, each taking $\text{poly}(|I|)$ time. Total time is $\text{poly}(|I|)$. □

3.3. Polynomial Balance Preservation

Theorem 4 (Instance Size Control)**.** *Let $I_0, I_1, \ldots, I_k$ be a sequence of instances constructed by Algorithm 1. If Conjecture 3 and 5 hold, then for all $j \in \{0, \ldots, k\}$:*

1. $|I_j| = \text{poly}(|I_0|)$.
2. *The transformation from I_j to I_{j+1} is computable in time* $\text{poly}(|I_j|)$.
3. *The number of steps $k = \text{poly}(|I_0|)$.*

Proof. We prove by induction on j.

Base case ($j = 0 \to j = 1$): By *Conjecture 5*, the initial reduction to a Tunnell instance has size $|I_1| = \text{poly}(|I_0|)$ and is computable in $\text{poly}(|I_0|)$ time.

Inductive step ($j \to j + 1$): Assume $|I_j| = \text{poly}(|I_0|)$. There are two types of transitions:

Type 1 (Tunnell halving $D_n \to C_n$): Same n, so $|I_{j+1}| = |I_j|$. Computation requires only division by 2, taking $O(1)$ time.

Type 2 (Finding new instance $C_n \to D_{n'}$): We need n' with:

- n' even square-free congruent,
- $|D_{n'}|$ polynomially smaller than $|C_n|$,
- $|n'| = \text{poly}(|n|)$.

By *Conjecture 3* with $B = \text{poly}(|I_j|)$ and target count $T = |C_n|/\text{poly}(|I_j|)$, such an n' exists and can be found in time $\text{poly}(|I_j|) = \text{poly}(|I_0|)$.

The bit length of n' satisfies:

$$\begin{aligned} |n'| &\leq |g| + O(1) \\ &= O(\log m + \log n + \log(m^2 - n^2)) \\ &= O(\log \text{poly}(|I_j|)) \\ &= \text{poly}(\log |I_j|) \\ &= \text{poly}(|I_0|). \end{aligned}$$

Thus $|I_{j+1}| = \text{poly}(|I_0|)$ and the transformation takes time $\text{poly}(|I_0|)$.

 doi:10.20944/preprints202511.1586.v1

Termination bound: Let S_j denote the solution count at step j. Each Type 2 transition ensures $S_{j+1} \leq S_j / \text{poly}(|I_0|)$. Since $S_j \geq 1$ for all $j < k$:

$$1 \leq S_k \leq S_0 \cdot [\text{poly}(|I_0|)]^{-\ell}$$

where ℓ is the number of Type 2 transitions. This gives $\ell \leq \log S_0 / \log \text{poly}(|I_0|) = \text{poly}(|I_0|)$ since $S_0 \leq 2^{\text{poly}(|I_0|)}$.

Including Type 1 transitions (at most one per Type 2), we have $k = O(\ell) = \text{poly}(|I_0|)$. □

4. The Main Conjectural Reduction

4.1. The Reduction Conjecture

Conjecture 5 (Reduction Conjecture)**.** *There exists a polynomial-time reduction R from any #P-complete problem to Tunnell instances such that:*

1. *For input instance I of a #P-complete problem, $R(I) = n$ where n is an even square-free congruent number.*
2. *The solution counts are polynomially related: $|D_n| = \Theta(\text{poly}(|I|) \cdot \#I)$, where $\#I$ denotes the solution count of I.*
3. *The reduction R is computable in time $\text{poly}(|I|)$ (under the assumption P=NP).*

Remark 2 (The Central Obstacle)**.** *This conjecture forms the heart of our proposed framework. A plausible construction pathway is as follows:*

- ***Step 1 (Diophantine encoding):*** *By Matiyasevich's theorem (Theorem 2), there exists a polynomial $\Phi(x_1, \ldots, x_k)$ such that satisfying assignments of φ correspond to integer solutions of $\Phi = 0$. Under P = NP, this encoding Φ can be constructed in polynomial time.*
- ***Step 2 (Instance aggregation):*** *Construct a single large integer m_I by aggregating the coefficients and structure of Φ (e.g., $m_I = \text{Encode}(\Phi)$).*
- ***Step 3 (Congruum construction):*** *Construct a single congruum g_I for the instance I, for example by setting $n = 1$ and $m = m_I$: $g_I = 4m_I \cdot 1 \cdot (m_I^2 - 1^2)$.*
- ***Step 4 (Square-free extraction):*** *Factor g_I (in P time under P=NP) and extract $n_I = \text{squarefree}(g_I / 2^e)$.*

*However, **Step 5**, the asserted solution count relationship, is a monumental, unproven leap.*

$$|D_{n_I}| = \Theta(\text{poly}(|I|) \cdot \#SAT(\varphi)) \quad \textbf{(This is the unproven gap)}$$

This conjecture requires that the entire logical structure of a Boolean formula φ can be embedded into a single integer n_I in such a way that the number of integer solutions (x, y, z) to the simple quadratic form $n_I = 8x^2 + 2y^2 + 16z^2$ parsimoniously reflects the number of satisfying assignments of φ.

While Matiyasevich's theorem provides a translation from NP to Diophantine solvability, it says nothing about preserving the number of solutions. Proving Conjecture 5 would require a completely new and profound connection between logic and the arithmetic of specific ternary quadratic forms. This is the main open problem proposed by this paper.

4.2. The Cascading Algorithm (Conditional)

If we *assume* our conjectures hold, we can define the following algorithm.

Theorem 6 (Conditional Algorithm Correctness)**.** If *Conjectures 3 and 5 are true, then under the assumptions P = NP and BSD, Algorithm 1 correctly computes the solution count of any #P-complete problem instance I in time* $\text{poly}(|I|)$.

Proof. Correctness: The logical flow of the algorithm is valid.

- Line 1: The initial mapping is correct by *assumption* of Conjecture 5.
- Lines 5–6: Tunnell's theorem guarantees $2C_n = D_n$ (by BSD assumption).
- Line 7: Finding a smaller instance is possible by *assumption* of Conjecture 3.

 doi:10.20944/preprints202511.1586.v1

- Lines 11–12: The base case and reconstruction are standard.

Termination: By Theorem 4 (which itself depends on the conjectures), the algorithm terminates in $\text{poly}(|I|)$ iterations.

Complexity: Each iteration takes $\text{poly}(|I|)$ time (by Lemma 2 and the assumed poly-time search from Conjecture 3). Total time is $\text{poly}(|I|)$. □

Algorithm 1 Count#P-Complete (Conditional)

Require: Instance I of #P-complete problem

Ensure: Solution count $\#I$

$n_0 \leftarrow \text{ReduceToTunnell}(I)$ {**Assumes Conjecture 5**}
Verify n_0 is even square-free congruent {BSD + P=NP}
$j \leftarrow 0$, $n_{\text{current}} \leftarrow n_0$
while $\text{SolutionCount}(n_{\text{current}}) > \text{poly}(|I|)$ **do**
  $\text{count}_D \leftarrow \text{Count}D(n_{\text{current}})$ {Lemma 2}
  $\text{count}_C \leftarrow \text{count}_D/2$ {Tunnell halving}
  $n_{\text{next}} \leftarrow \text{FindCongruentNumber}(\text{count}_C / \text{poly}(|I|), \text{poly}(|I|))$
  {**Assumes Conjecture 3**}
  Verify polynomial balance between n_{current} and n_{next}
  $n_{\text{current}} \leftarrow n_{\text{next}}$
  $j \leftarrow j + 1$
end while
$\text{base\_count} \leftarrow \text{DirectCount}(n_{\text{current}})$ {Small instance}
$\text{total} \leftarrow \text{base\_count} \times \text{ReconstructionFactor}(\text{cascade\_path})$
return total

4.3. Main Conditional Implication

Theorem 7 (Main Conditional Implication)**.** *If Conjectures 3 and 5 are true, then the following implication holds:*

$$(P = NP \wedge BSD) \implies (\#P = FP).$$

Proof. Assume Conjectures 3 and 5 are true. Assume P = NP and BSD also hold. By Conjecture 5, every #P-complete problem reduces to counting D_n. By Theorem 6, this counting problem is solvable in polynomial time (FP). Since a #P-complete problem is in FP, all of #P collapses to FP. □

Corollary 1 (Contrapositive Form)**.** *If $\#P \neq FP$ (widely believed), and if Conjectures 3 and 5 are true, then*

$$P \neq NP \quad \vee \quad BSD \text{ is false.}$$

Proof. This is the direct contrapositive of Theorem 7. □

5. Discussion

5.1. Implications for P Versus NP

Corollary 1 provides a novel, though highly conditional, perspective on the P versus NP question. It suggests that if one could prove the two number-theoretic conjectures (Conjectures 3 and 5), then the P vs. NP problem would be linked to the BSD conjecture.

 doi:10.20944/preprints202511.1586.v1

This framework transforms the problem. Instead of attacking P vs. NP directly, it suggests an alternative route:

1. Prove the Solution Density Conjecture (a hard problem in analytic number theory).
2. Prove the Reduction Conjecture (a seemingly monumental task in logic and arithmetic).
3. If successful, a major complexity-theoretic implication would follow.

5.2. The Role of Assumptions

Our framework requires P=NP and BSD as assumptions to get the conditional algorithm to run, ultimately leading to the '#P = FP' conclusion.

- **P = NP:** This assumption is the "engine" of the algorithm. It allows for polynomial-time factorization, binary search counting (Lemma 2), and the search procedure in Conjecture 3.
- **BSD:** This assumption is the "compass." It ensures Tunnell's theorem is a reliable two-way test ($2C_n = D_n \iff n$ is congruent), guaranteeing that the numbers we find and test in our algorithm are indeed the congruent numbers they are supposed to be.

The interdependence of these assumptions is subtle. One cannot simply replace P = NP with a weaker assumption, as the counting techniques fundamentally require NP = coNP = P.

5.3. Potential Weaknesses and Open Questions

The main "weaknesses" of this work are the two central conjectures, which are the main "open questions."

1. **Proving the Reduction Conjecture (5):** This is the primary obstacle. Is there any reason to believe a parsimonious reduction from #SAT to counting solutions to $n = 8x^2 + 2y^2 + 16z^2$ exists?
2. **Proving the Solution Density Conjecture (3):** This is a deep problem in analytic number theory, likely very difficult on its own.
3. **Dependence on specific curves:** We use curves of the form $y^2 = x^3 - n^2x$. Could a similar framework apply to other families of elliptic curves?
4. **Reverse implications:** Does #P = FP imply anything about BSD or our conjectures?

5.4. Relationship to Prior Work

The connection between number theory and complexity theory has been explored in various contexts [10,11]. However, to our knowledge, this is the first work to propose a framework linking BSD to the P versus NP question via counting complexity, contingent on these new conjectures.

Tunnell's theorem has been studied computationally [12], but not from a complexity-theoretic perspective. Our work suggests that if Conjecture 5 is true, the apparent "ease" of checking Tunnell's criterion (computing C_n and D_n) may hide deep complexity.

5.5. Comparison with Known Complexity Results

Table 1 summarizes how our *conjectural framework* relates to the complexity landscape. The key takeaway is that the problem "Counting Tunnell Solutions" is *conjectured* to be #P-complete.

Table 1. Complexity comparison of related problems

Problem	Known Complexity	Under P=NP+BSD+Conjectures
Deciding congruent numbers	Unknown	P
Counting Tunnell solutions	Unknown	#P-complete (by Conj. 5)
#SAT	#P-complete	P (FP)
Permanent	#P-complete	P (FP)
Factorization	Unknown, $\subseteq$ NP $\cap$ coNP	P

 doi:10.20944/preprints202511.1586.v1

6. Extended Proofs

6.1. Proof of Solution Count Preservation

We provide additional details for the solution count preservation through the cascade.

Lemma 3 (Solution Count Tracking). *Let $n_0, n_1, \ldots, n_k$ be the sequence of congruent numbers in Algorithm 1. If the conjectures hold, then the solution count of the original instance I can be recovered from the base case count via:*

$$\#I = \frac{D_{n_0}}{\text{poly}(|I|)} = \frac{D_{n_k} \cdot \prod_{j=0}^{k-1} r_j}{\text{poly}(|I|)}$$

where r_j is the reduction factor at step j.

Proof. At each Tunnell halving step $D_{n_j} \to C_{n_j}$, we have $C_{n_j} = D_{n_j}/2$ exactly.

At each instance transition $C_{n_j} \to D_{n_{j+1}}$, by construction (via Conjecture 3), we have:

$$D_{n_{j+1}} = \frac{C_{n_j}}{\text{poly}_j(|I|)}$$

for some polynomial function poly_j.

Combining these:

$$\begin{aligned} D_{n_k} &= \frac{C_{n_{k-1}}}{\text{poly}_{k-1}(|I|)} = \frac{D_{n_{k-1}}}{2 \cdot \text{poly}_{k-1}(|I|)} \\ &= \frac{D_{n_{k-2}}}{2^2 \cdot \text{poly}_{k-1}(|I|) \cdot \text{poly}_{k-2}(|I|)} \\ &= \cdots \\ &= \frac{D_{n_0}}{2^k \cdot \prod_{j=0}^{k-1} \text{poly}_j(|I|)}. \end{aligned}$$

Since $D_{n_0} = \Theta(\text{poly}(|I|) \cdot \#I)$ by Conjecture 5, and $\prod_{j=0}^{k-1} \text{poly}_j(|I|)$ is itself polynomial in $|I|$ (product of polynomially many polynomials), we can recover $\#I$ by the stated formula. □

6.2. Verification Under coNP

Lemma 4 (Cascade Verification). *Under NP = coNP = P, and assuming the conjectures hold, the entire cascade can be verified in polynomial time.*

Proof. For each step j, we need to verify:

1. n_j is square-free: Factor and check each prime has exponent 1. Time: $\text{poly}(|n_j|) = \text{poly}(|I|)$.
2. n_j is congruent: Compute $2C_{n_j}$ and D_{n_j}, check equality. Time: $\text{poly}(|n_j|)$ by Lemma 2.
3. Polynomial balance: Check $|D_{n_{j+1}}| \in [|C_{n_j}|/B^2, |C_{n_j}| \cdot B^2]$ for $B = \text{poly}(|I|)$. Time: $\text{poly}(|I|)$.
4. Size bound: Check $|n_j| \leq \text{poly}(|I|)$. Time: $O(1)$.

Total verification per step: $\text{poly}(|I|)$. Number of steps: $\text{poly}(|I|)$ by Theorem 4. Total time: $\text{poly}(|I|)$. □

7. Additional Remarks

7.1. Generalization to Other Elliptic Curves

While we focus on curves of the form $E_n : y^2 = x^3 - n^2x$, the framework may extend to other families. For instance, curves with rank growth governed by BSD could potentially yield similar complexity-theoretic implications. This remains an open direction.

 doi:10.20944/preprints202511.1586.v1

7.2. Quantum Computation

Under quantum computation models, factorization is in BQP (Shor's algorithm). However, our reduction still requires solving #P-complete problems, which remain hard even for quantum computers (unless BQP = #P, which is not believed). Thus, the main result persists in quantum settings with appropriate modifications.

7.3. Average-Case Complexity

Our results concern worst-case complexity. An interesting question is whether average-case hardness of #P problems similarly relates to average-case properties of BSD or congruent numbers.

8. Conclusion

We have proposed a conditional and conjectural framework connecting the P vs. NP problem, the Birch–Swinnerton-Dyer conjecture, and the #P-completeness of Tunnell's counting problem. We have shown that if two strong number-theoretic conjectures (the Solution Density Conjecture and the Reduction Conjecture) are true, then '$(\mathrm{P} = \mathrm{NP} \wedge \mathrm{BSD}) \implies (\#\mathrm{P} = \mathrm{FP})$'.

The core contribution of this paper is not a proof, but the formalization of these two conjectures, which themselves represent a new, potential line of attack. The central open question is no longer if the implication holds, but whether the required arithmetic reduction (Conjecture 5) exists at all. Proving this reduction would be a breakthrough of the highest order, linking the logical complexity of computation to the deepest structures of arithmetic geometry.

This work opens several avenues for future research, primarily:

1. Attempting to prove or disprove Conjecture 5.
2. Investigating the statistical properties of D_n to make progress on Conjecture 3.
3. Exploring whether other arithmetic problems (e.g., counting points on other varieties) could be more suitable targets for a #P-reduction.
4. Examining whether partial progress on BSD yields conditional complexity results.

Our results suggest deep, previously unexplored connections between arithmetic geometry and computational complexity theory. Whether these connections can be made unconditional, or provide insights into either P versus NP or BSD, remains an intriguing open question.

Acknowledgments: The author would like to thank Iris, Marilin, Sonia, Yoselin, and Arelis for their support.

Appendix A. Pseudocode Details

For completeness, we provide detailed pseudocode for the key subroutines. These algorithms are conditional on P=NP and our conjectures.

 doi:10.20944/preprints202511.1586.v1

Algorithm A1 ReduceToTunnell (Hypothetical)

Require: #P-complete instance I

Ensure: Even square-free congruent number n_I

{This function's existence is Conjecture 5}
Construct Diophantine encoding $\Phi(x_1, \ldots, x_k)$ of I {Matiyasevich}
coeffs $\leftarrow$ GetCoefficients(Φ)
structure $\leftarrow$ GetStructure(Φ)
$m_I \leftarrow$ AggregateInstance(coeffs, structure) {Encode Φ as a single integer}
$g_I \leftarrow 4 \cdot m_I \cdot (m_I^2 - 1)$ {Construct single congruum}
Factor g_I using polynomial-time factorization (under P=NP)
$e \leftarrow \max\{j : 2^j \mid g_I\}$
$n_I \leftarrow$ squarefree$(g_I/2^e)$
if n_I is not even **then**
    $n_I \leftarrow$ squarefree$((g_I/2^{e-1}))$ {Adjust to ensure evenness}
end if
{We conjecture $D_{n_I} = \Theta(\text{poly}(|I|) \cdot \#I)$}
return n_I

Algorithm A2 FindCongruentNumber (Hypothetical)

Require: Target count T, size bound B

Ensure: Even square-free congruent number n' with $D_{n'} \in [T/B^2, T \cdot B^2]$

{This function's success is Conjecture 3}
for $m = 2$ **to** B **do**
    for $n = 1$ **to** $m - 1$ **do**
        $g \leftarrow 4mn(m^2 - n^2)$
        $n' \leftarrow$ squarefree$(g/2^{\max\{j:2^j \mid g\}})$ {Extract square-free part}
        if n' is even **then**
            count $\leftarrow$ CountD(n') {Lemma 2}
            if count $\in [T/B^2, T \cdot B^2]$ **then**
                return n'
            end if
        end if
    end for
end for
return FAIL {Conjecture 3 asserts this does not happen}

 doi:10.20944/preprints202511.1586.v1

Algorithm A3 CountD (Binary Search Method)

Require: Even square-free congruent number n

Ensure: $D_n = \#\{(x, y, z) \in \mathbb{Z}^3 : n = 8x^2 + 2y^2 + 16z^2\}$

{This algorithm is in FP under P=NP}
low $\leftarrow 0$
high $\leftarrow 2^{\text{poly}(|n|)}$ {Upper bound on solutions}
while low $<$ high **do**
  mid $\leftarrow \lfloor (\text{low} + \text{high} + 1)/2 \rfloor$ {Use ceiling for binary search}
  {NP oracle, poly-time under P=NP}
  if VerifyAtLeast(n, mid) **then**
    low $\leftarrow$ mid
  else
    high $\leftarrow$ mid $- 1$
  end if
end while
return low

Algorithm A4 VerifyAtLeast

Require: Even square-free congruent number n, threshold k

Ensure: TRUE if $D_n \geq k$, FALSE otherwise

{This is an NP oracle, so in P by assumption P=NP}
Search for k distinct solutions (x_i, y_i, z_i) to $n = 8x^2 + 2y^2 + 16z^2$
if k distinct solutions found **then**
  return TRUE
else
  return FALSE
end if

References

1. Cook, S.A. The complexity of theorem-proving procedures. In Proceedings of the Proceedings of the Third Annual ACM Symposium on Theory of Computing, New York, NY, USA, 1971; pp. 151–158. https://doi.org/10.1145/3588287.3588297.
2. Karp, R.M. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*; Miller, R.E.; Thatcher, J.W.; Bohlinger, J.D., Eds.; Plenum: New York, USA, 1972; pp. 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9.
3. Birch, B.J.; Swinnerton-Dyer, H.P.F. Notes on elliptic curves. II. *Journal für die reine und angewandte Mathematik* **1965**, *218*, 79–108. https://doi.org/10.1515/crll.1965.218.79.
4. Tunnell, J.B. A classical Diophantine problem and modular forms of weight 3/2. *Inventiones Mathematicae* **1983**, *72*, 323–334. https://doi.org/10.1007/BF01389327.
5. Valiant, L.G. The complexity of computing the permanent. *Theoretical Computer Science* **1979**, *8*, 189–201. https://doi.org/10.1016/0304-3975(79)90044-6.
6. Arora, S.; Barak, B. *Computational Complexity: A Modern Approach*; Cambridge University Press: Cambridge, UK, 2009.

 doi:10.20944/preprints202511.1586.v1

7. Koblitz, N. *Introduction to Elliptic Curves and Modular Forms*, 2nd ed.; Vol. 97, *Graduate Texts in Mathematics*, Springer-Verlag: New York, 1993.
8. Dickson, L.E. *History of the Theory of Numbers, Vol. II: Diophantine Analysis*; Carnegie Institution of Washington: Washington, D.C., 1920.
9. Matiyasevich, Y.V. Enumerable sets are Diophantine. *Soviet Mathematics Doklady* **1970**, *11*, 354–358.
10. Adleman, L.; Huang, M.D. Function field sieve method for discrete logarithms over finite fields. *Information and Computation* **1999**, *151*, 5–16. https://doi.org/10.1006/inco.1998.2761.
11. Koiran, P. Hilbert's Nullstellensatz is in the polynomial hierarchy. *Journal of Complexity* **1996**, *12*, 273–286. https://doi.org/10.1006/jcom.1996.0019.
12. Cohen, H. *A Course in Computational Algebraic Number Theory*; Vol. 138, *Graduate Texts in Mathematics*, Springer-Verlag: Berlin, 1993.