Sharp-P and the Birch and Swinnerton-Dyer conjecture

Frank Vega
CopSonic, 1471 Route de Saint-Nauphary 82000 Montauban, France

Abstract

Assuming the Birch and Swinnerton-Dyer conjecture, an odd square-free integer n is a congruent number if and only if the number of triplets of integers (x, y, z) satisfying $2 \cdot x^2 + y^2 + 8 \cdot z^2 = n$ is twice the number of triplets satisfying $2 \cdot x^2 + y^2 + 32 \cdot z^2 = n$ due to Tunnell's theorem. However, we show these equations are instances of a variant of counting solutions of the homogeneous Diophantine equations of degree two which is a *#P–complete* problem. Deciding whether n is congruent or not is a problem in NP since congruent numbers could be easily checked by a congruum, because of every congruent number is a product of a congruum and the square of a rational number. We conjecture that if $P = NP$ and $FP \neq \#P$, then the Birch and Swinnerton-Dyer conjecture would be false.

2012 ACM Subject Classification Theory of computation Complexity classes; Theory of computation Problems, reductions and completeness

Keywords and phrases complexity classes, boolean formula, completeness, polynomial time

1 Introduction

Let $\{0, 1\}^*$ be the infinite set of binary strings, we say that a language $L_1 \subseteq \{0, 1\}^*$ is polynomial time reducible to a language $L_2 \subseteq \{0, 1\}^*$, written $L_1 \leq_p L_2$, if there is a polynomial time computable function $f : \{0, 1\}^* \to \{0, 1\}^*$ such that for all $x \in \{0, 1\}^*$:

$x \in L_1$ *if and only if* $f(x) \in L_2$.

An important complexity class is *NP–complete* [5]. If L_1 is a language such that $L' \leq_p L_1$ for some $L' \in$ *NP–complete*, then L_1 is *NP–hard* [2]. Moreover, if $L_1 \in NP$, then $L_1 \in$ *NP–complete* [2]. A principal *NP–complete* problem is SAT [5]. An instance of SAT is a Boolean formula ϕ which is composed of:

1. Boolean variables: $x_1, x_2, \ldots, x_n$;
2. Boolean connectives: Any Boolean function with one or two inputs and one output, such as $\wedge$(AND), $\vee$(OR), $\rightharpoondown$(NOT), $\Rightarrow$(implication), $\Leftrightarrow$(if and only if);
3. and parentheses.

A truth assignment for a Boolean formula ϕ is a set of values for the variables in ϕ. A satisfying truth assignment is a truth assignment that causes ϕ to be evaluated as true. A Boolean formula with a satisfying truth assignment is satisfiable. The problem SAT asks whether a given Boolean formula is satisfiable [5]. We define a CNF Boolean formula using the following terms:

A literal in a Boolean formula is an occurrence of a variable or its negation [2]. A Boolean formula is in conjunctive normal form, or CNF, if it is expressed as an AND of clauses, each of which is the OR of one or more literals [2]. A Boolean formula is in 3-conjunctive normal form or $3CNF$, if each clause has exactly three distinct literals [2]. For example, the Boolean formula:

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

is in $3CNF$. The first of its three clauses is $(x_1 \vee \rightharpoondown x_1 \vee \rightharpoondown x_2)$, which contains the three literals x_1, $\rightharpoondown x_1$, and $\rightharpoondown x_2$. In computer science, not-all-equal 3-satisfiability (*NAE–3SAT*) is an *NP–complete* variant of SAT over $3CNF$ Boolean formulas. *NAE–3SAT* consists in knowing whether a Boolean formula ϕ in $3CNF$ has a truth assignment such that for each clause at least one literal is true and at least one literal is false [5]. *NAE–3SAT* remains *NP–complete* when all clauses are monotone (meaning that variables are never negated), by Schaefer's dichotomy theorem [10].

In computational complexity, the complexity class $\#P$ (or Sharp-P) is the set of the counting problems associated with the decision problems in the set NP [12]. Besides, the complexity class FP is the set of the function problems associated with the decision problems in the set P [8]. Whether $FP = \#P$ or not is an open problem [8]. A problem is *#P–complete* if it is in $\#P$ and every $\#P$ problem has a Turing reduction or polynomial-time counting reduction to it. In some cases we use the parsimonious reductions which is a more specific type of reduction that preserves the exact number of solutions.

The counting version of *NAE–3SAT* on monotone clauses is *#P–complete* since to date, all known *NP–complete* languages have a defining relation which is *#P–complete* [7]. We know that the variant of $XOR\ 2SAT$ that uses the logic operator $\oplus$ (XOR) instead of $\vee$ (OR) within the clauses of $2CNF$ Boolean formulas can be decided in polynomial time [6, 9]. We announce a variant of its counting version which is in *#P–complete*.

▶ **Definition 1.** ***#Monotone Exact XOR 2SAT (#EX2SAT)***

INSTANCE: A Boolean formula φ in $2CNF$ with monotone clauses between logic operators $\oplus$ and a positive integer K.

ANSWER: Count the number of truth assignments in φ such that in each truth assignment there are exactly K satisfied clauses.

▶ **Theorem 2.** $\#EX2SAT \in$ *#P–complete.*

A homogeneous Diophantine equation is a Diophantine equation that is defined by a polynomial whose nonzero terms all have the same degree [3]. The degree of a term is the sum of the exponents of the variables that appear in it, and thus is a non-negative integer [3]. From general homogeneous Diophantine equations of degree two, we can reject an instance when there is no solution reducing the equation modulo p. We define another counting problem:

▶ **Definition 3.** ***#ZERO-ONE Homogeneous Diophantine Equation (#HDE)***

INSTANCE: A homogeneous Diophantine equation of degree two $P(x_1, x_2, \ldots, x_n) = B$ with the unknowns $x_1, x_2, \ldots, x_n$ and a positive integer B.

ANSWER: Count the number of solutions $u_1, u_2, \ldots, u_n$ on $\{0, 1\}^n$ where we have $P(x_1, x_2, \ldots, x_n) = B$.

▶ **Theorem 4.** $\#HDE \in$ *#P–complete.*

We generalize this problem.

▶ **Definition 5.** ***#Modulo Homogeneous Diophantine Equation (#MHDE)***

INSTANCE: A homogeneous Diophantine equation of degree two $P(x_1, x_2, \ldots, x_n) = B$ with the unknowns $x_1, x_2, \ldots, x_n$ and two positive integers B, M.

ANSWER: Count the number of solutions $u_1 \mod M, u_2 \mod M, \ldots, u_n \mod M$ on non-negative integers evaluated with modulo M such that $P(x_1, x_2, \ldots, x_n) = B$.

▶ **Theorem 6.** $\#MHDE \in$ *#P–complete.*

Proof. This is trivial since we can make a parsimonious reduction from $(P(x_1, x_2, \ldots, x_n), B)$ in $\#HDE$ to $(P(x_1, x_2, \ldots, x_n), B, 2)$ in $\#MHDE$ (i.e. using $M = 2$). Due to $\#HDE$ is in *#P–complete*, then $\#MHDE$ is in *#P–hard*. Finally, we know that $\#MHDE$ is in $\#P$. ◀

Assuming the Birch and Swinnerton-Dyer conjecture, an odd square-free integer n is a congruent number if and only if the number of triplets of integers (x, y, z) satisfying $2 \cdot x^2 + y^2 + 8 \cdot z^2 = n$ is twice the number of triplets satisfying $2 \cdot x^2 + y^2 + 32 \cdot z^2 = n$ due to Tunnell's theorem [11]. Deciding whether n is congruent or not is a problem in NP since congruent numbers could be easily checked by a congruum since every congruent number is a product of a congruum and the square of a rational number [1]. Certainly, every congruum is in the form of $4 \cdot m \cdot n \cdot (m^2 - n^2)$ (with $m > n$), where m and n are two distinct positive integers [4]. Thus, we state our finally conjecture:

▶ **Conjecture 7.** *Under the assumption that $P = NP$ and $FP \neq \#P$, then the Birch and Swinnerton-Dyer conjecture would be false.*

Proof. Under the assumption that $P = NP$, we know that deciding whether an odd square-free integer n is congruent or not can be done in polynomial time since this problem is in NP. On the other hand, for a given n, counting the numbers of solutions of $2 \cdot x^2 + y^2 + 8 \cdot z^2 = n$ and $2 \cdot x^2 + y^2 + 32 \cdot z^2 = n$ can be calculated by exhaustively searching through x, y, z in the range $-\sqrt{n}, \ldots, \sqrt{n}$. Note that, the solutions with negative values in x, y, z can be generated by the equivalent non-negative values. For example, if there is a solution in (u_x, u_y, u_z), then $(-u_x, u_y, u_z)$ is also a solution when $u_x \neq 0$ and so on. Hence, we can multiply the number of non-negative solutions by 8 and be able to obtain all the possible number of solutions for these equations. After that, we must subtract the exceeded amount of those non-negative triplets of integers (x, y, z) that contain a single or double zeros (once or two times, respectively) where the remaining values can be positive. We know the amount of triplets of integers (x, y, z) which contains a zero and the remaining values can be positive is not exponential and so, we could find them and count them in polynomial time under the assumption that $P = NP$. However, the instances $2 \cdot x^2 + y^2 + 8 \cdot z^2 = n$ and $2 \cdot x^2 + y^2 + 32 \cdot z^2 = n$ belong to the *#P–complete* problem $\#MHDE$ just using $B = n$ and $M = \lceil \sqrt{n} \rceil$, where $\lceil \ldots \rceil$ is the ceiling function when we consider only the non-negative values on the triplets. Since $FP \neq \#P$, then the problem $\#MHDE$ cannot be solved in polynomial time. We don't know specifically whether counting the number of non-negative integer solutions of the instances $2 \cdot x^2 + y^2 + 8 \cdot z^2 = n$ and $2 \cdot x^2 + y^2 + 32 \cdot z^2 = n$ cannot be solved in polynomial time as well. If that would be the case, then we might obtain a contradiction and therefore, the Birch and Swinnerton-Dyer conjecture would be false by reductio ad absurdum. ◀

2 Proof of Theorem 2

Proof. Take a Boolean formula ϕ in $3CNF$ with n variables and m clauses when all clauses are monotone. Iterate for each clause $c_i = (a \vee b \vee c)$ and create the conjunctive normal form formula

$$d_i = (a \oplus a_i) \wedge (b \oplus b_i) \wedge (c \oplus c_i) \wedge (a_i \oplus b_i) \wedge (a_i \oplus c_i) \wedge (b_i \oplus c_i)$$

where a_i, b_i, c_i are new variables linked to the clause c_i in ϕ. Note that, the clause c_i has exactly at least one true literal and at least one false literal if and only if d_i has exactly one unsatisfied clause. We notice that the value of positive literals a, b, c coincide in c_i and d_i, which means that those values are linked one-to-one in both directions. Finally, we obtain a new formula

$$\varphi = d_1 \wedge d_2 \wedge d_3 \wedge \ldots \wedge d_m$$

where there is not any repeated clause. In this way, we made a parsimonious reduction from ϕ in *#Monotone NAE–3SAT* to $(\varphi, 5 \cdot m)$ in $\#EX2SAT$. As we mentioned before,

#Monotone NAE–3SAT is in *#P–complete* and thus, $\#EX2SAT$ is in *#P–hard*. Moreover, we know that $\#EX2SAT$ is in *#P*. ◀

3 Proof of Theorem 4

Proof. Take a Boolean formula φ in $XOR\ 2CNF$ with n variables and m clauses when all clauses are monotone and a positive integer K. Iterate for each clause $c_i = (a \oplus b)$ and create the Homogeneous Diophantine Equation of degree two

$$P(x_a, x_b) = x_a^2 - 2 \cdot x_a \cdot x_b + x_b^2$$

where x_a, x_b are variables linked to the positive literals a, b in the Boolean formula φ. When the literals a, b are evaluated in $\{false, true\}$, then we assign the respective values $\{0, 1\}$ to the variables x_a, x_b (1 if it is true and 0 otherwise). Note that, the clause c_i is satisfied if and only if $P(x_a, x_b) = 1$. We notice that c_i is unsatisfied if and only if $P(x_a, x_b) = 0$, so the corresponding and translated values are linked one-to-one in both directions. Finally, we obtain a polynomial

$$P(x_1, x_2, \ldots, x_n) = P(x_a, x_b) + P(x_c, x_d) + \ldots + P(x_e, x_f)$$

that is a Homogeneous Diophantine Equation of degree two. Indeed, K satisfied clauses in φ correspond to K distinct small pieces of Homogeneous Diophantine Equation of degree two $P(x_i, x_j)$ which are equal to 1. In this way, we made a parsimonious reduction from (φ, K) in $\#EX2SAT$ to $(P(x_1, x_2, \ldots, x_n), K)$ in $\#HDE$. Since we obtain that $\#EX2SAT$ is in *#P–complete*, then $\#HDE$ is in *#P–hard*. Furthermore, we know that $\#HDE$ is in *#P*. ◀

References

1. Keith Conrad. The congruent number problem. *The Harvard College Mathematics Review*, 2(2):58–74, 2008.
2. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.
3. David A Cox, John Little, and Donal O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.
4. David Darling. *The universal book of mathematics from Abracadabra to Zeno's paradoxes*. John Wiley & Sons, Inc., 2004.
5. Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Company, 1 edition, 1979.
6. Neil D Jones, Y Edmund Lien, and William T Laaser. New problems complete for nondeterministic log space. *Mathematical systems theory*, 10(1):1–17, 1976. `doi:10.1007/BF01683259`.
7. Noam Livne. A note on #P-completeness of NP-witnessing relations. *Information processing letters*, 109(5):259–261, 2009. `doi:10.1016/j.ipl.2008.10.009`.
8. Christos H Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
9. Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):1–24, 2008. `doi:10.1145/1391289.1391291`.
10. Thomas J Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.
11. Jerrold B Tunnell. A classical Diophantine problem and modular forms of weight 3/2. *Inventiones mathematicae*, 72(2):323–334, 1983. `doi:10.1007/BF01389327`.
12. Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979. `doi:10.1016/0304-3975(79)90044-6`.