The Complexity of Goldbach's Conjecture

Frank Vega
Joysonic, Uzun Mirkova 5, Belgrade, 11000, Serbia
vega.frank@gmail.com

Abstract

The Goldbach's conjecture has been described as the most difficult problem in the history of Mathematics. This conjecture states that every even integer greater than 2 can be written as the sum of two primes. This is known as the strong Goldbach's conjecture. The conjecture that all odd numbers greater than 7 are the sum of three odd primes is known today as the weak Goldbach conjecture. A major complexity class is NSPACE(S(n)) for some S(n). We show if the weak Goldbach's conjecture is true, then the problem PRIMES is not in NSPACE(S(n)) for all S(n) = o(log n). However, if PRIMES is not in NSPACE(S(n)) for all S(n) = o(log n), then the strong Goldbach's conjecture is true or this has an infinite number of counterexamples. Since Harald Helfgott proved that the weak Goldbach's conjecture is true, then the strong Goldbach's conjecture is true or this has an infinite number of counterexamples, where the contains of infinite number of counterexamples is statistically less unlike it.

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness; Mathematics of computing → Number-theoretic computations

Keywords and phrases complexity classes, number theory, primes, regular languages

1 Introduction

Number theory is a branch of pure mathematics devoted primarily to the study of the integers and integer-valued functions [15]. Goldbach's conjecture is one of the most important and unsolved problems in number theory [8]. Nowadays, it is one of the open problems of Hilbert and Landau [8]. Goldbach's original conjecture, written on 7 June 1742 in a letter to Leonhard Euler, states: "... at least it seems that every number that is greater than 2 is the sum of three primes" [6]. This is known as the ternary Goldbach conjecture. We call a prime as a natural number that is greater than 1 and has exactly two divisors, 1 and the number itself [18]. However, the mathematician Christian Goldbach considered 1 as a prime number. Euler replied in a letter dated 30 June 1742 the following statement: "Every even integer greater than 2 can be written as the sum of two primes" [6]. This is known as the strong Goldbach conjecture.

Using Vinogradov's method [17], it has been showed that almost all even numbers can be written as the sum of two primes. In 1973, Chen showed that every sufficiently large even number can be written as the sum of some prime number and a semiprime [4]. The strong Goldbach conjecture implies the conjecture that all odd numbers greater than 7 are the sum of three odd primes, which is known today as the weak Goldbach conjecture [6]. In 2012 and 2013, Peruvian mathematician Harald Helfgott published a pair of papers claiming to improve major and minor arc estimates sufficiently to unconditionally prove the weak Goldbach conjecture [9], [10].

We define a non-regular language under the assumption that the weak Goldbach's conjecture is true. Indeed, the possible language can be easily proved that might be non-regular using the Pumping lemma for regular languages [13]. However, we prove this language is actually regular when the exponentially more succinct version is in $NSPACE(S(n))$ for some $S(n) = o(\log n)$. This result is based on the breakthrough approach that checking whether a number is prime can be decided in polynomial time by a deterministic Turing machine [1].

This problem is known as *PRIMES* [1]. Since the weak Goldbach's conjecture is true, then we obtain that necessarily the mentioned language should be non-regular [9], [10]. Nevertheless, this implies the language *PRIMES* is not in $NSPACE(S(n))$ for all $S(n) = o(\log n)$. Moreover, if the language *PRIMES* is not in $NSPACE(S(n))$ for all $S(n) = o(\log n)$, then the strong Goldbach's conjecture is true or this has an infinite number of counterexamples. In this way, we prove the strong Goldbach's conjecture is true or this has an infinite number of counterexamples.

2 Background Theory

In 1936, Turing developed his theoretical computational model [16]. The deterministic and nondeterministic Turing machines have become in two of the most important definitions related to this theoretical model for computation [16]. A deterministic Turing machine has only one next action for each step defined in its program or transition function [16]. A nondeterministic Turing machine could contain more than one action defined for each step of its program, where this one is no longer a function, but a relation [16].

Let Σ be a finite alphabet with at least two elements, and let Σ^* be the set of finite strings over Σ [3]. A Turing machine M has an associated input alphabet Σ [3]. For each string w in Σ^* there is a computation associated with M on input w [3]. We say that M accepts w if this computation terminates in the accepting state, that is $M(w) = \text{“}yes\text{”}$ [3]. Note that M fails to accept w either if this computation ends in the rejecting state, that is $M(w) = \text{“}no\text{”}$, or if the computation fails to terminate, or the computation ends in the halting state with some output, that is $M(w) = y$ (when M outputs the string y on the input w) [3].

Another relevant advance in the last century has been the definition of a complexity class. A language over an alphabet is any set of strings made up of symbols from that alphabet [5]. A complexity class is a set of problems, which are represented as a language, grouped by measures such as the running time, memory, etc [5]. The language accepted by a Turing machine M, denoted $L(M)$, has an associated alphabet Σ and is defined by:

$$L(M) = \{w \in \Sigma^* : M(w) = \text{“}yes\text{”}\}.$$

Moreover, $L(M)$ is decided by M, when $w \notin L(M)$ if and only if $M(w) = \text{“}no\text{”}$ [5]. We denote by $t_M(w)$ the number of steps in the computation of M on input w [3]. For $n \in \mathbb{N}$ we denote by $T_M(n)$ the worst case run time of M; that is:

$$T_M(n) = max\{t_M(w) : w \in \Sigma^n\}$$

where Σ^n is the set of all strings over Σ of length n [3]. We say that M runs in polynomial time if there is a constant k such that for all n, $T_M(n) \leq n^k + k$ [3]. In other words, this means the language $L(M)$ can be decided by the Turing machine M in polynomial time. Therefore, P is the complexity class of languages that can be decided by deterministic Turing machines in polynomial time [5].

A logarithmic space Turing machine has a read-only input tape, a write-only output tape, and read/write work tapes [16]. The work tapes may contain at most $O(\log n)$ symbols [16]. In computational complexity theory, NL is the complexity class containing the decision problems that can be decided by a nondeterministic logarithmic space Turing machine [12].

We use o-notation to denote an upper bound that is not asymptotically tight. We formally define $o(g(n))$ as the set

$$o(g(n)) = \{f(n) : \textit{ for any positive constant } c > 0, \textit{ there exists a constant}$$

$n_0 > 0$ *such that* $0 \leq f(n) < c \times g(n)$ *for all* $n \geq n_0\}$.

For example, $2 \times n = o(n^2)$, but $2 \times n^2 \neq o(n^2)$ [5].

In theoretical computer science and formal language theory, a regular language is a formal language that can be expressed using a regular expression [2]. The complexity class that contains all the regular languages is REG. The complexity class $NSPACE(f(n))$ is the set of decision problems that can be solved by a nondeterministic Turing machine M, using space $f(n)$, where n is the length of the input [11].

3 Results

▶ **Definition 1.** *We define the weak Goldbach's language* L_{WG} *as follows:*

$$L_{WG} = \{1^{2\times n+1}0^p0^q0^r : n \in \mathbb{N} \wedge n \geq 4 \wedge p, q \textit{ and } r \textit{ are odd primes } \wedge 2\times n+1 = p+q+r\}.$$

We define the strong Goldbach's language L_G *as follows:*

$$L_G = \{1^{2\times n}0^p0^q : n \in \mathbb{N} \wedge n \geq 3 \wedge p \textit{ and } q \textit{ are odd primes } \wedge 2 \times n = p + q\}.$$

▶ **Theorem 2.** *If the weak Goldbach's conjecture is true, then the weak Goldbach's language* L_{WG} *is non-regular. Moreover, if the strong Goldbach's conjecture is true, then the strong Goldbach's language* L_G *is non-regular.*

Proof. If the weak Goldbach's conjecture is true, then the weak Goldbach's language L_{WG} is equal to the another language L' defined as follows:

$$L' = \{1^{2\times n+1}0^{2\times n+1} : n \in \mathbb{N} \wedge n \geq 4\}.$$

We can easily prove that L' is non-regular using the Pumping lemma for regular languages [13]. Moreover, if the strong Goldbach's conjecture is true, then the strong Goldbach's language L_G is equal to the another language L'' defined as follows:

$$L'' = \{1^{2\times n}0^{2\times n} : n \in \mathbb{N} \wedge n \geq 3\}.$$

We can easily prove that L'' is non-regular using the Pumping lemma for regular languages as well [13]. ◀

▶ **Definition 3.** *We define the weak verification Goldbach's language* L_{WVG} *as follows:*

$$L_{WVG} = \{(2\times n+1, p, q, r) : n \in \mathbb{N} \wedge n \geq 4 \wedge p, q \textit{ and } r \textit{ are odd primes } \wedge 2\times n+1 = p+q+r\}.$$

We define the strong verification Goldbach's language L_{VG} *as follows:*

$$L_{VG} = \{(2 \times n, p, q) : n \in \mathbb{N} \wedge n \geq 3 \wedge p \textit{ and } q \textit{ are odd primes } \wedge 2 \times n = p + q\}.$$

▶ **Theorem 4.** $L_{WVG} \in P$ *and* $L_{VG} \in P$.

Proof. This result is based on the breakthrough approach that checking whether a number is prime can be decided in polynomial time by a deterministic Turing machine [1]. Certainly, we can check in polynomial time whether p, q and r are odd primes and the other verifications can be easily done in polynomial time as well. ◀

▶ **Definition 5.** *We define the weak Goldbach's language with separator L_{WSG} as follows:*

$$L_{WSG} = \{0^{2\times n+1}\#0^p\#0^q\#0^r : n \in \mathbb{N} \wedge n \geq 4 \wedge p, q \text{ and } r \text{ are odd primes } \wedge 2\times n+1 = p+q+r\}$$

and we define the strong Goldbach's language with separator L_{SG} as follows:

$$L_{SG} = \{0^{2\times n}\#0^p\#0^q : n \in \mathbb{N} \wedge n \geq 3 \wedge p \text{ and } q \text{ are odd primes } \wedge 2\times n = p+q\}$$

where # is the blank symbol.

▶ **Lemma 6.** *The weak Goldbach's language with separator L_{WSG} is the unary representation of the weak verification Goldbach's language L_{WVG}. The strong Goldbach's language with separator L_{SG} is the unary representation of the strong verification Goldbach's language L_{VG}.*

Proof. This is trivially true from the definition of these languages. ◀

▶ **Theorem 7.** *If $L_{WVG} \in$ NSPACE$(S(n))$ for some $S(n) = o(\log n)$, then $L_{WG} \in$ REG.*

Proof. In case of $L_{WVG} \in$ *NSPACE*$(S(n))$ for some $S(n) = o(\log n)$, then there is a nondeterministic Turing machine which decides L_{WSG} that uses space that is smaller than $c \times \log\log n$ for all $c > 0$, because of L_{WSG} is the unary version of L_{WVG} due to Lemma 6 [7]. Certainly, the standard space translation between the unary and binary languages actually works for nondeterministic machines with small space [7]. This means that if some language belongs to *NSPACE*$(S(n))$, then the unary version of that language belongs to *NSPACE*$(S(\log n))$ [7]. In this way, we obtain that $L_{WSG} \in$ *REG* because of *REG* = *NSPACE*$(o(\log\log n))$ [11]. In addition, we can reduce in a nondeterministic constant space the language L_{WG} to L_{WSG} just nondeterministically inserting the blank symbol # within two arbitrary positions between the 0's on the input. Moreover, this nondeterminism reduction inserts the blank symbol # between the 1's and 0's and converts the 1's to 0's from the original input of L_{WG} just generating the final output to L_{WSG}. Consequently, we prove $L_{WG} \in$ *REG* under the assumption that $L_{WVG} \in$ *NSPACE*$(S(n))$ for some $S(n) = o(\log n)$, since *REG* is also the complexity class of languages decided by nondeterministic Turing machines in constant space [14]. ◀

▶ **Theorem 8.** *$L_{WVG} \notin$ NSPACE$(S(n))$ for all $S(n) = o(\log n)$.*

Proof. If the weak Goldbach's conjecture is true, then $L_{WG} \notin$ *REG* as a consequence of Theorem 2. However, if $L_{WVG} \in$ *NSPACE*$(S(n))$ for some $S(n) = o(\log n)$, then $L_{WG} \in$ *REG* due to Theorem 7. In this way, the weak Goldbach's conjecture cannot be true under the assumption that $L_{WVG} \in$ *NSPACE*$(S(n))$ for some $S(n) = o(\log n)$. Since the weak Goldbach's conjecture is true, then we obtain that $L_{WVG} \notin$ *NSPACE*$(S(n))$ for all $S(n) = o(\log n)$ [9], [10]. ◀

▶ **Theorem 9.** *$PRIMES \notin$ NSPACE$(S(n))$ for all $S(n) = o(\log n)$.*

Proof. From the Theorem 8, we obtain that $L_{WVG} \notin$ *NSPACE*$(S(n))$ for all $S(n) = o(\log n)$. However, the checking of whether the four numbers on the input are odds and proving the equality of the sum can be done in *NSPACE*$(\log\log n)$. Certainly, the verification of the odd property could be done in constant space. In addition, the verification of the equality of the sum $2 \times n + 1 = p + q + r$ can be done in *NSPACE*$(\log\log n)$, since we need a constant space to save the remainder of the sum from each step and a binary string of length bounded by $\log\log n$ which represents the position of the bits that we are currently summing. For example, if we want to check whether the binary numbers 1, 10000001, 100000001 and 110000011 comply with the sum $110000011 = 1 + 10000001 + 100000001$, then we start for the rightmost one until the leftmost ones using the binary digit 1 as a remainder only in the first step and saving the position of the bits we are summing using at most the binary number 1001, because of 9 is the greatest bit position. The ultimate remaining verification that we need to analyze in L_{WVG} is whether p, q and r are primes. Since $\log\log n = o(\log n)$ and $L_{WVG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$, then we have as unique remaining possibility that $PRIMES \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. ◀

▶ **Theorem 10.** *The strong Goldbach's conjecture is true or this has an infinite number of counterexamples.*

Proof. If the strong Goldbach's conjecture is false, then $L_G \in REG$ or L_G is non-regular and its complement is infinite, since every finite set is regular [12]. However, this implies that the exponentially more succinct version of L_G, that is L_{VG}, should be in $NSPACE(S(n))$ for some $S(n) = o(\log n)$, because of $REG = NSPACE(o(\log\log n))$ and the same algorithm that decides L_G in $NSPACE(o(\log\log n))$ could be easily transformed into a slightly modified algorithm that decides L_{VG} in $NSPACE(S(n))$ for some $S(n) = o(\log n)$ [11]. However, that is not possible because of $L_{VG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$ when $PRIMES \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. Certainly, the verification of whether p and q are primes need to be done in order to accept the elements of this language. Consequently, we obtain that $L_G \notin REG$, since it is not possible that $L_G \in NSPACE(o(\log\log n))$ under the result that $L_{VG} \notin NSPACE(S(n))$ for all $S(n) = o(\log n)$. In this way, we obtain a contradiction just assuming that the strong Goldbach's conjecture is false and $L_G \in REG$. In contraposition, we have the strong Goldbach's conjecture is true or this has an infinite number of counterexamples. ◀

References

1 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of Mathematics*, 160(2):781–793, 2004. doi:10.4007/annals.2004.160.781.

2 Alfred V. Aho and John E. Hopcroft. *The Design and Analysis of Computer Algorithms*. Pearson Education India, 1974.

3 Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

4 Jing-run Chen. On the Representation of a Large Even Integer as the Sum of a Prime and the Product of Two Primes at Most. *Sci. Sinica*, 16:157–176, 1973.

5 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3 edition, 2009.

6 Leonard Eugene Dickson. *History of the Theory of Numbers: Divisibility and Primality*, volume 1. New York, Dover, 2005.

7 Viliam Geffert and Dana Pardubská. Unary Coded NP-Complete Languages in ASPACE (log log n). *International Journal of Foundations of Computer Science*, 24(07):1167–1182, 2013. doi:10.1007/978-3-642-31653-1_16.

8 Richard K. Guy. *Unsolved Problems in Number Theory*. New York, Springer-Verlag, 3 edition, 2004.

9 Harald A. Helfgott. Minor arcs for Goldbach's problem. *arXiv preprint arXiv:1205.5252*, 2012.

10 Harald A. Helfgott. Major arcs for Goldbach's theorem. *arXiv preprint arXiv:1305.2897*, 2013.

11 Pascal Michel. A survey of space complexity. *Theoretical computer science*, 101(1):99–132, 1992. doi:10.1016/0304-3975(92)90151-5.

12 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

13 Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM journal of research and development*, 3(2):114–125, 1959. `doi:10.1147/rd.32.0114`.

14 John C. Shepherdson. The Reduction of Two-Way Automata to One-Way Automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959. `doi:10.1147/rd.32.0198`.

15 Joseph H. Silverman. *A Friendly Introduction to Number Theory.* Pearson Education, Inc., 4 edition, 2012.

16 Michael Sipser. *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston, 2006.

17 Ivan M. Vinogradov. Representation of an Odd Number as a Sum of Three Primes. *Comptes Rendus (Doklady) de l'Académie des Sciences de l'USSR*, 15:169–172, 1937.

18 David Wells. *Prime Numbers, The Most Mysterious Figures in Math.* John Wiley & Sons, Inc., 2005.