On the Hardness of PosSLP

Peter Bürgisser[*] Gorav Jindal[†]

Abstract

The problem PosSLP involves determining whether an integer computed by a given straight-line program is positive. This problem has attracted considerable attention within the field of computational complexity as it provides a complete characterization of the complexity associated with numerical computation. However, non-trivial lower bounds for PosSLP remain unknown. In this paper, we demonstrate that PosSLP $\in$ BPP would imply that $\mathsf{NP} \subseteq \mathsf{BPP}$, under the assumption of a conjecture concerning the complexity of the radical of a polynomial proposed by Dutta, Saxena, and Sinhababu (STOC'2018). Our proof builds upon the established NP-hardness of determining if a univariate polynomial computed by an SLP has a real root, as demonstrated by Perrucci and Sabia (JDA'2005).

1 Introduction

1.1 Straight-line Programs Given an integer a as input, how do we decide whether a is positive or negative? This question seems very innocuous at the first glance. Indeed, if a is given as a bit string, the question is trivial. This question becomes interesting when we are given a compact expression for a instead of its bit string representation. One such compact way to represent an integer is by an arithmetic circuit or, equivalently, a straight-line program. These are fundamental concepts studied in algebraic complexity theory: we refer the reader to excellent surveys [SY10, Sap21].

An *arithmetic circuit* is a directed acyclic graph, whose leaves are labeled by formal variables $x_1, \ldots, x_n$ or scalars from the underlying field $\mathbb{F}$. The non-leaf nodes are arithmetic gates. We assume that there is a unique output node and use the term gate for node interchangeably. Every gate of such a circuit computes a multivariate polynomial in the canonical way. The polynomial computed at the output gate is said to be the polynomial computed by the circuit. The size of the circuit is defined as the number of gates in it. We shall restrict our attention to constant free arithmetic circuits, which compute univariate polynomials. We define a *straight-line program P, SLP* for short, to be a sequence of univariate integer polynomials $(a_0, a_1, \ldots, a_\ell)$ such that $a_0 = 1, a_1 = x$ and $a_i = a_j \circ a_k$ for all $2 \leq i \leq \ell$, where $\circ \in \{+, -, *\}$ and $j, k < i$. We say that P computes the univariate polynomial a_ℓ and that P has length ℓ. Note that in this definition, we do not allow any constants different from 1. For an integer univariate polynomial $f \in \mathbb{Z}[x]$, we define $\tau(f)$ as the length of the smallest SLP which computes f. It is clear that any SLP of length ℓ can be described using $O(\ell \log \ell)$ bits. Since an integer is a special case of a univariate polynomial, SLPs can also compute integers.

1.2 PosSLP and Related Work Now we formally define the *problem* PosSLP, introduced in [ABKPM09], which is the central object of study of this paper.

Problem 1.1. (PosSLP) *Given an SLP P computing an integer n_P, decide if $n_P > 0$.*

This problem was introduced to establish a connection between classical models of computation and the *Blum-Shub-Smale model* [BCSS97]. The latter is an extensively studied model for studying computations with real numbers. A BSS machine M runs according to a finite program and takes as input a finite sequence of real numbers of arbitrary length, i.e., an element of $\cup_n \mathbb{R}^n$. Moreover, M has an infinite tape consisting of cells containing real numbers or blanks. In each step, M can copy the content of one cell into another, perform an arithmetic operation $\circ \in \{+, -, \times, \div\}$ on two cells, or branch by comparing any cell to 0. The class $\mathsf{P}^0_{\mathbb{R}}$ denotes the

[*] Institut für Mathematik, Technische UniversitätBerlin, Berlin, Germany. Email: pbuerg@math.tu-berlin.de.

[†] Max Planck Institute for Software Systems, Saarbrücken, Germany. Part of the work was done while author was a member of Graduiertenkolleg "Facets of Complexity/Facetten der Komplexität" (GRK 2434) and Institut für Mathematik, Technische Universität Berlin. Email: gorav.jindal@gmail.com.

set of decision problems decided by polynomial time by *constant free* BSS machines. To compare this with classical complexity classes, defined via Turing machines, one considers the Boolean part $\mathsf{BP}(\mathsf{P}^0_{\mathbb{R}}) := \{L \cap \{0,1\}^n \mid L \in \mathsf{P}^0_{\mathbb{R}}\}$.

In [ABKPM09], it was shown that the computational power of this complexity class is given by polynomial time computations with oracle calls to PosSLP. That is:

Proposition 1.1. $\mathsf{P}^{\text{PosSLP}} = \mathsf{BP}(\mathsf{P}^0_{\mathbb{R}})$.

[ABKPM09] also explained the relevance of PosSLP for numerical computation in a more direct way, without referring to the formal model of BSS machines, as follows. For any nonzero real number r, we can write $r = s2^m$ with $\frac{1}{2} \le |s| < 1$ and $m \in \mathbb{Z}$. A floating point approximation of r with k significant bits is a floating point number $t2^m$ such that $|s - t| \le 2^{-(k+1)}$.

Problem 1.2. (The generic task of numerical computation) *Given an arithmetic circuit C computing a polynomial $f(x_1, \ldots, x_n)$, given floating point numbers $a_1, \ldots, a_n$, and an integer k in unary, along with a promise that $f(a_1, \ldots, a_n)$ is nonzero, compute a floating point approximation of the value of the output $f(a_1, \ldots, a_n)$ with k significant bits.*

[ABKPM09] showed that Problem 1.2 is polynomial time Turing equivalent to PosSLP. This assertion and Proposition 1.1 support the hypothesis that PosSLP does not have efficient algorithms. A related problem DegSLP, introduced in [ABKPM09], is the problem of computing the degree of a polynomial given as input by an arithmetic circuit. [ABKPM09] also showed that DegSLP reduces to PosSLP. This further suggests the computational intractability of PosSLP, given the belief that an efficient algorithm for DegSLP is unlikely to exist.

As for upper bounds on PosSLP, the following result in terms of the counting hierarchy [Wag86, Kon09, AB09] is the best known result.

Theorem 1.1. ([ABKPM09]) PosSLP $\in \mathsf{P}^{\mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}}$.

Jindal and Saranurak [JS12] observed that if monotone SLP complexity τ_+ and SLP complexity τ of positive integers are polynomially equivalent, then PosSLP $\in \Sigma^{\mathsf{P}}_2 \subseteq \mathsf{PH}$. There are several other important problems which reduce to PosSLP. One such well-studied and important problem is the following.

Problem 1.3. (Sum-of-square-roots problem, SSR) *Given a list $(a_1, \ldots, a_n)$ of positive integers and a positive integer k, decide if $\sum_{i \in [n]} \sqrt{a_i} \ge k$.*

This is asked as an open problem in [GGJ76]. It has connections to the Euclidean traveling salesman problem. The Euclidean traveling salesman problem is not known to be in NP, but is readily seen to be in NP relative to an SSR oracle. By using classical Newton iteration, SSR reduces to PosSLP [ABKPM09]. The sum-of-square-roots problem was conjectured to be in P in [Mal01].

Another important problem is to decide the inequality of succinctly represented integers [ESY14]. More precisely, consider the following problem.

Problem 1.4. (Inequality testing of succinctly represented integers) *Given positive integers $a_1, \ldots, a_n$, $b_1, \ldots, b_n, c_1, \ldots, c_m, d_1, \ldots, d_m$, decide if $\prod_{i=1}^n a_i^{b_i} \ge \prod_{i=1}^m c_i^{d_i}$.*

This problem is easily seen to be a special case of PosSLP. It was shown in [ESY14] that Problem 1.4 can be solved in deterministic polynomial time if one can prove strong lower bounds on integer linear combinations of logarithms of positive integers, known as the Lang-Waldschmidt conjecture [Lan13, Wal04], see [ESY14, Conjecture 3.2]. However, Theorem 1.1 still provides the best unconditional upper bound for Problem 1.4.

1.3 Our Results

Despite the non-trivial, but rather inefficient upper bound of Theorem 1.1, no strong hardness results are known for PosSLP. In terms of lower bounds, the only known connection is that Arithmetic Circuit Identity Testing (ACIT) reduces to PosSLP [ABKPM09]. In this paper, we show PosSLP $\in \mathsf{BPP}$ would have dramatic consequences for complexity theory, assuming a variant of the radical conjecture proposed in [DSS22].

We define the *radical* $\text{rad}(f)$ of a nonzero integer polynomial $f \in \mathbb{Z}[x_1, \ldots, x_n]$ as the product of the irreducible integer polynomials dividing f. It is also called the *square-free part* of f. Note that $\text{rad}(f)$ is uniquely defined up to a sign. This generalizes the radical of a nonzero integer n, which is defined as the product of the distinct prime numbers dividing n.

We shall crucially rely on the following conjecture, which is a constructive variant of the radical conjecture proposed in [DSS22]. See Section 5 for a discussion on Conjecture 1.1.

Conjecture 1.1. (Constructive univariate radical conjecture) *For any polynomial $f \in \mathbb{Z}[x]$, we have $\tau(\mathsf{rad}(f)) \leq \mathsf{poly}(\tau(f))$. Moreover, there is a randomized polynomial time algorithm which, given an SLP of size s computing f, constructs an SLP for $\mathsf{rad}(f)$ of size $\mathsf{poly}(s)$ with success probability at least $1 - \frac{1}{\Omega(s^{1+\epsilon})}$ for some $\epsilon > 0$.*

In fact, for our purposes, it is enough to know that Conjecture 1.1 applies to some nonzero integer multiple of $\mathsf{rad}(f)$. The following result is the main contribution of this paper; see Section 3 for the proof.

Theorem 1.2. *If Conjecture 1.1 is true and* PosSLP $\in$ BPP *then* NP $\subseteq$ BPP.

We also show that counting the real roots of univariate polynomials computed by straight-line programs is #P-hard. For a univariate polynomial F, let us denote by $Z_{\mathbb{R}}(F)$ the number of its real roots, counted with multiplicity. Consider the problem:

Problem 1.5. (CountRealRoots) *Given a SLP P computing a univariate polynomial $f_P(x) \in \mathbb{Z}[x]$, compute $Z_{\mathbb{R}}(f_P)$.*

We prove the following result in Section 4.

Theorem 1.3. CountRealRoots *is #P-hard.*

1.4 Proof Ideas We rely on the proof of NP-hardness due to [PS07] of the following problem. The idea for this reduction goes back to [Pla84], see Section 2 for a detailed discussion.

Problem 1.6. *Given an SLP P computing a univariate integer polynomial, decide whether this polynomial has a real root.*

Proof idea for Theorem 1.2: The reduction in [PS07] computes for a given 3SAT formula W in polynomial time an SLP computing a univariate polynomial $P \in \mathbb{Z}[x]$ such that W is satisfiable iff P has a real root. All the real roots of P are in the interval $[-1, 1]$. Moreover, every real root of P (if any) has multiplicity two, and P never attains negative values. Now we call on Conjecture 1.1 to construct an SLP computing the radical $R := \mathsf{rad}(P)$. By the definition of the radical, the real roots of R are exactly the real roots of P, but with multiplicity one. If W is not satisfiable, then P has no real roots, and therefore, neither does R. In this case, R does not change signs on $[-1, 1]$: either it completely remains below the x-axis or completely remains above the x-axis. On the other hand, if W is satisfiable, then R does cross the x-axis at the real roots of P, because every root of R has multiplicity exactly one. Hence it attains both negative and positive values. Therefore:

- If W is not satisfiable, then R does not change sign on the interval $[-1, 1]$.
- If W is satisfiable, then R attains both negative and positive values on $[-1, 1]$.

Now we sample a random rational point a from the interval $[-1, 1]$. By oracle calls to PosSLP, we can compute the sign of R at a and 1. By construction, if W is not satisfiable, then $R(a)$ has the same sign as $R(1)$. But if W is satisfiable, then there are choices of a for which $R(a)$ and $R(1)$ have different signs. Our novel contribution is that we can adapt the NP-hardness reduction in [PS07] such that $R(a)$ and $R(1)$ have different signs with a constant probability, for a random $a \in [-1, 1]$, provided W is satisfiable. To this end, we will assume that W has at most one satisfying assignment, which can be achieved using the randomized polynomial time reduction of 3SAT to Unique-SAT [VV86]. Under this assumption, the set of real roots of R has a simpler structure, which allows us to prove that $R(a)$ and $R(1)$ have different signs with constant probability for a random $a \in [-1, 1]$, see Section 3 for details.

Proof idea for Theorem 1.3: We use the ideas developed in Section 4 of [vzGKS96] and the reduction of [PS07] outlined above. Let us denote by $\#W$ the number of satisfying assignments of a 3SAT formula W. It is well known that computing $\#W$ is $\#\mathsf{P}$-complete. The strategy is to prove that $\#W$ can be computed in polynomial time, if oracle calls to CountRealRoots are allowed.

For a given 3SAT formula W, we compute the polynomial P from above, which has a real root iff W is satisfiable. It turns out that P has $N(\phi)$ many roots for each satisfying assignment ϕ of W, where $N(\phi) := \prod_{p_i \notin \phi}(p_i - 1)$. Here ϕ is seen as a subset of a set of n odd primes $p_1, \dots, p_n$, see Section 2. In the reduction of [PS07], one can choose any odd primes p_i. Here we first choose an odd prime q, and then the prime p_i is chosen from the arithmetic progression $\{aq + 2 \mid a \in \mathbb{N}\}$. This can be done efficiently because primes in arithmetic progressions have sufficiently high density [BMOR18]. This implies that $N(\phi) \equiv 1 \bmod q$. We show that as a consequence, $\#W \equiv \frac{1}{2} Z_{\mathbb{R}}(P) \bmod q$. Therefore, by using oracle calls to CountRealRoots, we can compute $\#W \bmod q$ for any odd prime q. Doing so for sufficiently many odd primes q, and using Chinese remaindering, we can finally compute $\#W$. We refer to Section 4 for details.

2 Preliminaries

In this section we mainly recall the reduction from [PS07, Pla84] to prove that Problem 1.6 is NP-hard. For a positive integer n we write $[n] := \{1, 2, \dots, n\}$.

Let us first note the following folklore result for later use. It follows from the observation that the graph of a real polynomial crosses the x-axis only on the roots of odd multiplicity.

Lemma 2.1. *Let $a, b \in \mathbb{R}$ with $a < b$ and f is a real univariate polynomial. Assume $f(a)f(b) \neq 0$. Then f has an even number of real roots (counted with multiplicity) in (a, b) if and only if $f(a)f(b) > 0$.*

We also note the following easy observation for later use.

Lemma 2.2. *Consider the following problem: given a univariate polynomial $F(x)$ as an SLP and $p, q \in \mathbb{Z}$, compute the sign of $F(p/q)$. This problem reduces to* PosSLP *under polynomial time many one reductions.*

Proof. First notice that we have SLPs of length $O(\log p)$ and $O(\log q)$, which compute p and q, respectively. Suppose $F(x)$ is given by an SLP P of length ℓ. By induction on the length of SLPs, we show that we can efficiently construct from P two SLPs P_1, P_2 of length $O(\log p + \log q + \ell)$, computing integers r, s respectively, such that $F(p/q) = r/s$. Moreover $\operatorname{sgn}(r/s) = \operatorname{sgn}(rs)$. □

2.1 Chebychev polynomials The Chebychev polynomials T_k are univariate polynomials in one variable x defined by $T_0(x) := 1$, $T_1(x) := x$ and for an integer $k \geq 2$ by the recursion

$$T_k(x) := 2xT_{k-1}(x) - T_{k-2}(x).$$

Clearly, the T_k are integer polynomials. They have the following well known properties; see [KC91].

1. $\deg(T_k) = k$ and the leading coefficient of T_k is 2^{k-1}.

2. $T_k(x) = \cos(k \arccos(x))$ for all $x \in [-1, 1]$.

3. The roots of T_k are $\left\{\cos\left(t\frac{\pi}{2k}\right) \mid t \in \{1, 3, \dots, 2k-1\}\right\}$.

4. For every $p, q \in \mathbb{N}$, we have $T_p \circ T_q = T_{pq}$, where $\circ$ denotes the composition.

The following is an easy consequence of the properties of Chebychev polynomials.

Lemma 2.3. (Lemma 1 in [PS07]) *The Chebychev polynomial T_k can be computed by an SLP of length $O(k)$. Moreover, if $k = pq$ for $p, q \in \mathbb{N}$, then $T_k = T_{pq}$ can be computed by a straight-line program of length $O(p + q)$.*

2.2 Real Roots of Univariate Polynomials and Straight-line Programs We recall here the reduction [PS07, Pla84] from the well-known NP-complete problem 3SAT to Problem 1.6. The idea is to associate with the n literals $x_1, \ldots, x_n$ of a 3SAT formula once and for all n distinct odd primes $p_1, \ldots, p_n$. It will be convenient to abbreviate $p_{\max} := \max_i p_i$ and $p_{\min} := \min_i p_i$. We put $M := \prod_{i \in [n]} p_i$ and enumerate the roots of the M^{th} Chebychev polynomial T_M by the odd integers $t \in \mathrm{Odd}(M) := \{1, 3, \ldots, 2M-1\}$. Thus we define

$$r_M(t) := \cos\left(t \frac{\pi}{2M}\right) \tag{2.1}$$

and denote by $R_M := \{r_M(t) \mid t \in \mathrm{Odd}(M)\}$ the set of zeros of $T_M(x)$. This defines the bijection $\mathrm{Odd}(M) \to R_M, t \mapsto r_M(t)$ whose inverse we denote by $r \mapsto t_M(r)$.

We write $X := \{p_1, \ldots, p_n\}$ and identify subsets $\phi \subseteq X$ with Boolean assignments to the literals $x_1, \ldots, x_n$. More specifically, x_i is assigned "true" if and only if $p_i \in \phi$. We now consider the map

$$A_M : R_M \to \{0,1\}^X, \quad r \mapsto \{p_j \mid p_j \text{ divides } t_M(r)\},$$

which assigns to a root r of T_M the set of prime divisors of $t_M(r)$. Note that the function A_M is surjective but not injective. In fact, the fiber of A_M over $\phi \in \{0,1\}^X$ is given by

$$S_M(\phi) := A_M^{-1}(\phi) = \{r \in R_M \mid \gcd(t_M(r), M) = \alpha(\phi)\}, \tag{2.2}$$

where we have set

$$\alpha(\phi) := \prod_{p \in \phi} p.$$

Finally, we assign to a 3SAT formula W over the literals $x_1, \ldots, x_n$ the union of the sets $S_M(\phi)$, taken over all satisfying Boolean assignments, that is,

$$S_M(W) := \bigcup_{\phi \in \{0,1\}^X,\ \phi \text{ satisfies } W} S_M(\phi).$$

We denote by $\mathrm{PolySAT}_M(W)$ the *monic* univariate real polynomial with the set of roots $S_M(W)$:

$$\mathrm{PolySAT}_M(W) := \prod_{r \in S_M(W)} (x - r).$$

Note that $\mathrm{PolySAT}_M(W)$ is square free and only has real roots because it is a factor $T_M(x)$. As in [PS07], we can express $\mathrm{PolySAT}_M(W)$ in terms of the following analogues of the cyclotomic polynomials:

$$C_\ell(x) := \prod_{t \in \mathrm{Odd}(\ell),\ \gcd(t,\ell)=1} (x - r_\ell(t)). \tag{2.3}$$

If ℓ is odd, then the degree of C_ℓ is given by the Euler totient function (not to be confused with assignments ϕ)

$$\deg C_\ell = \varphi(\ell). \tag{2.4}$$

Equation (2.2) implies that for an assignment $\psi \in \{0,1\}^X$, we have:

$$\prod_{r \in S_M(\psi)} (x - r) = C_{M/\alpha(\psi)}(x). \tag{2.5}$$

This immediately implies that

$$\mathrm{PolySAT}_M(W) = \prod_{\psi \in \{0,1\}^X,\ \psi \text{ satisfies } W} C_{M/\alpha(\psi)}(x). \tag{2.6}$$

Note that all the integers $M/\alpha(\psi)$ are odd. Therefore, using Equation (2.4), we see that the number of real roots of $\mathrm{PolySAT}_M(W)$ is given by

$$Z_{\mathbb{R}}(\mathrm{PolySAT}_M(W)) = \sum_{\psi \text{ satisfies } W} \varphi(M/\alpha(\psi)). \tag{2.7}$$

The following properties of $\mathrm{PolySAT}_M(W)$ are easy to verify (see Lemma 5 in [PS07]). Note that $\frac{1}{2^{M-1}} T_M(x)$ is the monic polynomial obtained by dividing $T_M(x)$ by its leading coefficient.

Lemma 2.4. *Suppose W, W_1, W_2 are 3SAT formulas on the literals $x_1, \dots, x_n$, and the primes $p_1, \dots, p_n$ and $M = \prod_i p_i$ are as before. Then we have:*

1. *For a literal x_i, $\operatorname{PolySAT}_M(x_i) = \frac{T_{M/p_i}}{2^{M/p_i - 1}}$.*
2. *W_1 and W_2 are equivalent iff $\operatorname{PolySAT}_M(W_1) = \operatorname{PolySAT}_M(W_2)$.*
3. *$\operatorname{PolySAT}_M(\neg W) = \frac{T_M}{2^{M-1} \operatorname{PolySAT}_M(W)}$.*
4. *$\operatorname{PolySAT}_M(W_1 \wedge W_2) = \gcd(\operatorname{PolySAT}_M(W_1), \operatorname{PolySAT}_M(W_2))$.*
5. *$\operatorname{PolySAT}_M(W_1 \vee W_2) = \operatorname{lcm}(\operatorname{PolySAT}_M(W_1), \operatorname{PolySAT}_M(W_2))$.*

This lemma implies that $\operatorname{PolySAT}_M(W)$ has rational coefficients. We also recall Lemma 6 in [PS07], which says that the composition of $\operatorname{PolySAT}_M(W)$ with the Chebychev polynomial T_q, up to a scaling factor, equals $\operatorname{PolySAT}_{Mq}(W)$.

Lemma 2.5. *Let W be a 3SAT formula over literals $x_1, \dots, x_n$ and y be a new literal. Let q be a new odd prime associated to the literal y. If we think of W being a 3SAT formula over the literals $x_1, \dots, x_n, y$, then we have, for some $\lambda \in \mathbb{Q}^*$,*

$$\operatorname{PolySAT}_M(W) \circ T_q = \lambda \operatorname{PolySAT}_{Mq}(W).$$

We are now concerned with the efficient computation of $\operatorname{PolySAT}_M(W)$. Recall $p_{\max} = \max_i p_i$.

Lemma 2.6. *Let C be a clause formed by 3 literals x_i, x_j, x_k (and their negations). In time $O(p_{\max}^9)$, we can construct an SLP computing a nonzero integer polynomial $F_M(C)$, such that*

$$F_M(C) = I_M(C) \operatorname{PolySAT}_M(C)$$

for some nonzero integer $I_M(C)$.

Proof. Put $N := p_i p_j p_k$. Tracing the proof of Proposition 4 in [PS07], we see that in time $O(N^3)$, we can construct an SLP of size $O(N^3)$ computing a polynomial F, which is a nonzero integer multiple of $\operatorname{PolySAT}_N(C)$. From Lemma 2.5 we deduce that for some $\lambda \in \mathbb{Q}^*$,

$$\operatorname{PolySAT}_N(C) \circ T_{M/N} = \lambda \operatorname{PolySAT}_M(C).$$

After multiplying with a suitable integer, we can write this as $F_M(C) = I_M(C) \operatorname{PolySAT}_M(C)$ with some nonzero integer $I_M(C)$. With Lemma 2.3 we refer that $F_M(C)$ has an SLP of size $O(N^3 + np_{\max})$, which can be constructed in the same amount of time. To complete the proof, notice that $O(N^3 + np_{\max}) = O(p_{\max}^9)$. ∎

Theorem 2.1. *Suppose $W = C_1 \wedge \cdots \wedge C_m$ is a 3SAT formula on n literals with m clauses. Let $F_M(C_i)$ be the integer polynomial (multiple of $\operatorname{PolySAT}_M(C_i)$) constructed for the clause C_i in Lemma 2.6. Then:*

1. *An SLP computing the polynomial $P_M(W)$, defined as sum of squares,*

   $$P_M(W) := \sum_{i \in [m]} (F_M(C_i))^2,$$

   can be computed in time $O(mp_{\max}^9)$ for given W (and from the primes $p_1, \dots, p_n$).
2. *The polynomial $P_M(W)$ has the same set of real roots as $\operatorname{PolySAT}_M(W)$.*
3. *Every real root of $P_M(W)$ has multiplicity two.*
4. *The radical of $P_M(W)$ satisfies*

   $$\operatorname{rad}(P_M(W)) = \operatorname{PolySAT}_M(W) \cdot Q_M(W),$$

   where $Q_M(W)$ is an integer polynomial having no real roots.

Proof. 1. The first assertion follows from Lemma 2.6.

2. A real number r is a root of $P_M(W)$ if and only if r is a root of all the $F_M(C_i)$. The common zero set of $P_M(C_1), \dots, P_M(C_m)$ equals the zero set of $\mathrm{PolySAT}_M(W)$, since by Lemma 2.4

$$\mathrm{PolySAT}_M(W) = \gcd(\mathrm{PolySAT}_M(C_1), \mathrm{PolySAT}_M(C_2), \dots, \mathrm{PolySAT}_M(C_m)).$$

Also note that $P_M(C_i)$ only has real roots. It follows that $P_M(W)$ and $\mathrm{PolySAT}_M(W)$ have same set of real roots, which show the second assertion.

3. From the construction of $P_M(W)$, it follows that every real root of $\mathrm{PolySAT}_M(W)$ has multiplicity two.

4. Let us write $\mathrm{PolySAT}_M(W) = \prod_{i=1}^{a}(x-r_i)$. Then the factorization of $P_M(W)$ into irreducible polynomials in $\mathbb{Z}[x]$ has the form

$$P_M(W) = I \cdot \prod_{i=1}^{a}(x-r_i)^2 \cdot \prod_{j=1}^{b} h_j^{e_j},$$

where the irreducible polynomials h_j have no real roots, $e_j \geq 1$, and $I \in \mathbb{Z}$. Therefore

$$\mathrm{rad}(P_M(W)) = \mathrm{rad}(I) \prod_{i=1}^{a}(x-r_i) \cdot \prod_{j=1}^{b} h_j,$$

which is the fourth assertion with $Q_M(W) := \mathrm{rad}(I) \prod_{j=1}^{b} h_j$. □

Using the above construction, the following was derived in [PS07]. We provide the proof since our argument will be a refinement of it.

Theorem 2.2. *Problem 1.6 is* NP*-hard.*

Proof. Suppose W is a 3SAT formula on n literals. Using Lemma 2.6 and Theorem 2.1, in time $\mathsf{poly}(p_{\max}, m)$, we can construct an SLP, which computes a polynomial f that has same real roots as $\mathrm{PolySAT}_M(W)$, albeit with multiplicity two. By definition, $\mathrm{PolySAT}_M(W)$ has a real root if and only if W is satisfiable. Hence f has a real root if and only if W is satisfiable. By the well-known prime number theorem [HWHB$^+$08], $p_{\max}$ can be chosen to be of magnitude $O(n \log n)$. This proves that Problem 1.6 is NP-hard. □

3 NP-hardness of PosSLP

The following problem is crucial for showing the NP-hardness of PosSLP.

Problem 3.1. (Unique-SAT) *Given a 3SAT formula W with the promise that W has at most one satisfying assignment, decide if W is satisfiable.*

The following is well known.

Theorem 3.1. (Valiant-Vazirani, [VV86]) *There is a probabilistic polynomial time algorithm, which given a 3SAT formula W on n literals, outputs a 3SAT formula W' such that if W is satisfiable, then W' has a unique satisfying assignment with probability at least $\frac{1}{8n}$. If W is not satisfiable, then W' is also not satisfiable.*

We now explain how to use Conjecture 1.1 in the setting of Theorem 2.1. This is the key step, which makes our result conditional.

Corollary 3.1. *Suppose W is a 3SAT formula as in Theorem 2.1. If Conjecture 1.1 is true, then in randomized $\mathsf{poly}(p_{\max}, m)$ time, we can construct an SLP of size $\mathsf{poly}(p_{\max}, m)$, which computes a nonzero multiple $F_M(W) = \mathrm{PolySAT}_M(W) \cdot Q_M(W)$ of $\mathrm{PolySAT}_M(W)$, where the polynomial $Q_M(W)$ does not have real roots. The success probability of this randomized algorithm is at least $1 - \frac{1}{\Omega(n^{1+\epsilon})}$.*

Proof. By using Theorem 2.1, we know that $\mathrm{rad}(P_M(W)) = \mathrm{PolySAT}_M(W) \cdot Q_M(W)$, where Q_W is an integer polynomial having no real roots. Theorem 2.1 also shows that $P_M(W))$ has an SLP of size $O(mp_{\max}^9)$ and this SLP can also be constructed in time $O(mp_{\max}^9)$. By using Conjecture 1.1, in randomized $\mathsf{poly}(p_{\max}, m)$ time, we can construct an SLP of size $\mathsf{poly}(p_{\max}, m)$, which computes $\mathrm{rad}(P_M(W))$. By renaming $\mathrm{rad}(P_M(W))$ to $F_M(W)$, we obtain the desired claim. Since $\tau(P_M(W)) > n$, we get that the success probability is at least $1 - \frac{1}{\Omega(n^{1+\epsilon})}$.
□

By Theorem 3.1 we may assume that if a given 3SAT formula W is satisfiable, then it has at most one satisfying assignment. Hence we assume in this section that W has exactly one satisfying assignment ϕ. This assumption implies a simpler structure on the roots of $\mathrm{PolySAT}_M(W)$, as seen below in Lemma 3.1. Recall that $\alpha(\phi)$ is defined as $\alpha(\phi) := \prod_{p_i \in \phi} p_i$.

Lemma 3.1. *Suppose a 3SAT formula W over n literals has a unique satisfying assignment ϕ. Then, writing $N := \frac{M}{\alpha(\phi)}$, we have*

$$\mathrm{PolySAT}_M(W) = C_N(x) = \prod_{t \in \mathrm{Odd}(N),\ \gcd(t,N)=1} (x - r_N(t)) = \prod_{\gcd(t,2N)=1} (x - r_N(t)).$$

Proof. This is an immediate consequence of Equation (2.3) and Equation (2.6). □

We slightly modify the choice of the primes $p_1, \ldots, p_n$ in the reduction sketched in Section 2 to ensure that $p_{\min} \geq n^3$. Moreover, without loss of generality, we may assume that the unique satisfying assignment ϕ in Lemma 3.1 (if it exists) assigns at least one literal x_i to be *false*. This is possible since we can first check if the *all true assignment*, where all literals are true, satisfies W or not. This entails that $\alpha(\phi) \leq \frac{M}{p_{\min}}$, hence we can assume that

$$N \geq p_{\min} \geq n^3. \tag{3.8}$$

We think of the multiplicative group $\mathbb{Z}_{2N}^{\times}$ as the subset of $[2N]$ in decreasing order

$$\mathbb{Z}_{2N}^{\times} := \{t \in [2N] \mid \gcd(t, 2N) = 1\} = \{t_1, \ldots, t_{\varphi(N)}\},$$

where $t_j > t_{j+1}$, where $t_1 = 2N - 1$ and $t_{\varphi(N)} = 1$. The reason for this choice of indexing is that $t \mapsto r_N(t)$ is monotonically decreasing, so that we obtain $r_N(t_j) < r_N(t_{j+1})$, see Equation (2.1).

We can then rewrite Lemma 3.1 as

$$\mathrm{PolySAT}_M(W) = \prod_{j=1}^{\varphi(N)} (x - r_N(t_j)).$$

The roots of $\mathrm{PolySAT}_M(W)$ subdivide the interval $(-1, 1)$ into a collection $\mathcal{I} := \{I_0, I_1, \ldots, I_{\varphi(N)}\}$ of open intervals. More precisely, we define

$$I_j := \big(r_N(t_j), r_N(t_{j+1})\big) \quad \text{for } 1 \leq j < \varphi(N),$$

and $I_0 := (-1, r_N(2N-1))$, $I_{\varphi(N)} := (r_N(1), 1)$.

Suppose $F_M(W)$ is the polynomial constructed in Corollary 3.1. Since the real roots of $F_M(W)$ are exactly that of $\mathrm{PolySAT}_M(W)$, and all these roots are of multiplicity one, $F_M(W)$ does not change sign on the interval I_0. Without loss of generality, we assume that $F_M(W)$ is positive on the leftmost interval I_0. Using Lemma 2.1 we then infer that

$$F_M(W) \text{ is negative on the interval } I_j \in \mathcal{I} \text{ if and only if } j \text{ is odd.}$$

From now on, to simplify notation, we drop the index M and argument W in this section and just write $F := F_M(W)$. The above discussion implies that if we pick any real number a from any interval I_j in $\mathcal{I}$ of odd index j, then $F(a)$ and $F(1)$ have different signs, i.e., $F(a)$ is negative. An interval $I_j \in \mathcal{I}$ is said to be odd-indexed if j is odd, otherwise it is said to be even-indexed. Now our overall strategy can be summarized as follows (for a formal argument see Lemma 3.4):

1. Show that the sum of the lengths of odd-indexed intervals $I_j \in \mathcal{I}$ is at least c for some positive constant c.

2. Pick a "random" rational number a in the interval $(-1, 1)$. With probability at least $c/2$, we have $a \in I_j$ with j being odd.

3. Compute the sign of $F(a)F(1)$ using PosSLP. If $F(a)F(1) < 0$ then $\mathrm{PolySAT}_M(W)$ has a real root and hence W is satisfiable. This succeeds with probability at least $c/2$ if W is satisfiable.

Out next goal is to show the following result.

Proposition 3.1. *If W is satisfiable, then the sum of the lengths of odd-indexed intervals in $\mathcal{I}$ is at least $\frac{1}{\pi}$.*

For this, we rely on the two technical lemmas below, whose proof is postponed to Section 3.1. We call a subinterval J of $[-1,1]$ a *simple interval* if it connects two subsequent real roots of the Chebychev polynomial T_N. This means that $J = ((r_N(s_1), r_N(s_2))$ for $s_1, s_2 \in \mathrm{Odd}(N)$ with $s_1 - s_2 = 2$. Note that the end points of a simple such interval are not required to be zeros of $\mathrm{PolySAT}_M(W)$.

The first lemma states that a substantial part of the interval $(-1,1)$ is subsumed by simple intervals in $\mathcal{I}$.

Lemma 3.2. *The sum of the lengths of non-simple intervals in $\mathcal{I}$ is at most $\frac{1}{n}$.*

We also need that two adjacent simple intervals cannot differ too much in their lengths.

Lemma 3.3. *Suppose I, J are two adjacent simple intervals (not necessarily in $\mathcal{I}$). Then we have:*

$$\frac{1}{\pi} \leq \frac{\mathrm{length}(I)}{\mathrm{length}(J)} \leq \pi.$$

Proof. [Proof of Proposition 3.1] We denote by ℓ_{se} the sum of the lengths of simple even-indexed intervals in $\mathcal{I}$. Analogously, ℓ_{so} denotes the sum of the lengths of simple odd-indexed intervals in $\mathcal{I}$, and we write ℓ_o for the sum of the lengths of *all* odd-indexed intervals in $\mathcal{I}$. By Lemma 3.2 we have $2 \leq \ell_{se} + \ell_{so} + \frac{1}{n} \leq \ell_{se} + \ell_o + \frac{1}{n}$, hence $\ell_{se} \geq 2 - \ell_o - \frac{1}{n}$.

Consider a simple even-indexed interval $I_j \in \mathcal{I}$. The interval I_{j+1} contains a simple interval J. Therefore $|I_{j+1}| \,/\, |I_j| \geq |J| \,/\, |I_j| \geq \pi^{-1}$, where the right inequality is due to Lemma 3.3. Applying this argument to all simple even-indexed intervals in $\mathcal{I}$, and adding their length, we obtain that:

$$\ell_o \geq \frac{1}{\pi}\ell_{se} \geq \frac{1}{\pi}\big(2 - \ell_o - \frac{1}{n}\big).$$

A simple calculation shows that the above equation implies $\ell_o \geq \frac{1}{\pi+1}\left(2 - \frac{1}{n}\right) \geq \frac{1}{\pi}$ for $n > 1$. □

Lemma 3.4. *Assume $M \geq 3$. Suppose K is an integer in $[-M^4, M^4] \cap \mathbb{Z}$ chosen uniformly at random. If W is satisfiable, then we have $\mathrm{Prob}_K[F(K/M^4)F(1) < 0] \geq \frac{1}{4\pi}$. On the other hand, we always have $F(K/M^4)F(1) > 0$ if W is not satisfiable.*

Proof. If suffices to consider the case where W is satisfiable. We can think of sampling a point a in the set of grid points $\Gamma := [-1,1] \cap \frac{1}{M^4}\mathbb{Z}$, which has cardinality $2M^4 + 1$. Recall that $F(a)F(1) < 0$ implies that a lies in an odd-index interval $I_j \in \mathcal{I}$.

We claim that for $I_j \in \mathcal{I}$ with odd index j

$$|I_j \cap \Gamma| \geq M^4\,\mathrm{length}(I_j) - 2.$$

Indeed, assume $I_j = (a,b)$ and let $a' \geq a$ be minimal such that $a'M^4 \in \mathbb{Z}$. Similarly, let $b' \leq b$ be maximal such that $b'M^4 \in \mathbb{Z}$. Then $a' - a \leq M^{-4}$ and $b - b' \leq M^{-4}$. We obtain

$$|I_j \cap \Gamma| = (b' - a')M^4 + 1 \geq (b-a)M^4 - 2,$$

which shows the claim.

Summing over all odd-indexed intervals I_j in $\mathcal{I}$, we get

$$\sum_j |I_j \cap \Gamma| \geq M^4 \sum_j \mathrm{length}(I_j) - 2M \geq \frac{M^4}{\pi} - 2M,$$

where we used Proposition 3.1 for the right-hand inequality (here we use that W is satisfiable).

We can lower bound the probability that a uniformly random point $a \in \Gamma$ lands in an odd-indexed interval $I_j \in \mathcal{I}$ as follows (let $M \geq 3$) :

$$\frac{1}{|\Gamma|}\sum_j |I_j \cap \Gamma| \geq \frac{1}{2M^4+1}\Big(\frac{M^4}{\pi} - 2M\Big) \geq \frac{1}{4\pi},$$

where the right-hand inequality follows from a simple calculation. This completes the proof □

We now restate the main result of this paper.

THEOREM 3.2. *If Conjecture 1.1 is true and* PosSLP $\in$ BPP *then* $\mathsf{NP} \subseteq \mathsf{BPP}$.

Proof. Given a 3SAT formula W, we use Theorem 3.1 to compute a 3SAT formula W' such that if W is satisfiable, then W' has a unique satisfying assignment with probability at least $\frac{1}{8n}$. If W is not satisfiable then W' is also not satisfiable. Now we use Corollary 3.1 to compute an SLP which computes $F = F_M(W') = \mathrm{rad}(P_M(W'))$. Then we randomly sample $K \in [-M^4, M^4] \cap \mathbb{Z}$ and compute the sign of $(-1)F(K/M^4)F(1)$ based on Lemma 2.2.

By using the assumption PosSLP $\in$ BPP and amplifying the probability using standard probability amplification techniques, we assume that success probability of a PosSLP oracle is at least $1 - 2^{-n}$, see [AB09] or more specifically [Sud07, Definition 1].

Using Lemma 3.4 on $F = F_M(W')$, we see that if W is satisfiable, then $(-1)F(K/M^4)F(1) > 0$ happens with probability at least $\frac{1}{4\pi} \cdot \frac{1}{8n} \cdot \left(1 - \frac{1}{\Omega(n^{1+\epsilon})}\right) = \Omega\left(\frac{1}{n}\right)$. Hence the PosSLP oracle verifies the inequality $(-1)F(K/M^4)F(1) > 0$ with probability at least $\Omega\left(\frac{1}{n}\right) \cdot (1 - 2^{-n}) = \Omega\left(\frac{1}{n}\right)$.

On the other hand if W is not satisfiable, then neither is W'. Hence $(-1)F(K/M^4)F(1)$ can only be positive if $F \neq \mathrm{rad}(P_M(W'))$, Corollary 3.1 implies that this happens with probability at most $O\left(\frac{1}{n^{1+\epsilon}}\right)$. Hence $(-1)F(K/M^4)F(1) > 0$ happens with probability at most $O\left(\frac{1}{n^{1+\epsilon}}\right)$. Hence the PosSLP oracle verifies the inequality $(-1)F(K/M^4)F(1) > 0$ with probability at most $O\left(\frac{1}{n^{1+\epsilon}}\right) + 2^{-n} = O\left(\frac{1}{n^{1+\epsilon}}\right)$. After doing $\mathsf{poly}(n)$ many independent runs of this reduction, we can boost the success probability for both cases to $O(1)$ using standard probability amplification techniques, see [AB09] or more specifically [Sud07, Theorem 3]. Altogether, this shows that the 3SAT problem lies in BPP. This implies that $\mathsf{NP} \subseteq \mathsf{BPP}$.

□

3.1 Proofs of Technical Lemmas

Proof. [Proof of Lemma 3.2] There are two kind of non-simple intervals in $\mathcal{I}$. The first kind are the leftmost and the rightmost intervals: namely $I_0 = \left(-1, \cos\left(\frac{(2N-1)\pi}{2N}\right)\right)$ and $I_{\varphi(N)} = \left(\cos\left(\frac{\pi}{2N}\right), 1\right)$. We use the inequality $1 - \cos(x) \leq \frac{x^2}{2}$ to infer that the length of the interval I_0 is at most $\frac{\pi^2}{8N^2}$. Using a similar argument, the length of interval $I_{\varphi(N)}$ can also be upper bounded by $\frac{\pi^2}{8N^2}$. Hence the total length of both these intervals is at most $\frac{\pi^2}{4N^2}$.

The second kind of non-simple intervals in $\mathcal{I}$ are of the form:

$$(r_N(s_1)), r_N(s_2)) = \left(\cos\left(s_1 \frac{\pi}{2N}\right), \cos\left(s_2 \frac{\pi}{2N}\right)\right) \text{ with } s_1, s_2 \in \mathrm{Odd}(N) \text{ and } (s_1 - s_2) \geq 4. \tag{3.9}$$

Now we bound the length of these second kind of non-simple intervals. To this end we define: $D := \{r_N(t) \mid t \in \mathrm{Odd}(N) \setminus \mathbb{Z}_{2N}^{\times}\}$. The elements of D are the roots of T_N which are not the roots of $\mathrm{PolySAT}_M(W) = C_N$, since the root set of C_N is exactly $\{r_N(t) \mid t \in \mathbb{Z}_{2N}^{\times}\}$. We have $|D| = N - \varphi(N)$. Suppose $I = ((r_N(s_1)), r_N(s_2)) \in \mathcal{I}$ is a non-simple interval as in Equation (3.9). Since I is non-simple, its endpoints are not subsequent roots of T_N. Hence I contains some roots of T_N which are not the roots of C_N. More precisely, there are exactly $\frac{s_1 - s_2}{2} - 1$ elements of D in I. The length of such an interval I can be bounded as:

$$\mathrm{length}(I) = \cos\left(s_2 \frac{\pi}{2N}\right) - \cos\left(s_1 \frac{\pi}{2N}\right) = 2\sin\left(\frac{(s_1 + s_2)\pi}{4N}\right)\sin\left(\frac{(s_1 - s_2)\pi}{4N}\right) \leq \frac{(s_1 - s_2)\pi}{2N}.$$

Since $(s_1 - s_2) \geq 4$ implies that $(s_1 - s_2) \leq 2(s_1 - s_2) - 4$, we get that the length of such an I can be bounded as:

$$\mathrm{length}(I) \leq \frac{(s_1 - s_2)\pi}{2N} \leq \frac{2\pi}{N}\left(\frac{s_1 - s_2}{2} - 1\right).$$

Hence if a non-simple interval I contains m elements of D, then we have:

$$\mathrm{length}(I) \leq \frac{2\pi}{N} m.$$

Since $|D| = N - \varphi(N)$, we get that the total length of such non-simple intervals is at most $\frac{2\pi(N - \varphi(N))}{N}$. This proves that the total length of non-simple intervals is at most $\frac{2\pi(N - \varphi(N))}{N} + \frac{\pi^2}{4N^2}$. Now we use the inequality:

$$1 - x \geq e^{-2x} \text{ for all } 0 \leq x \leq \frac{1}{2} \tag{3.10}$$

to first obtain that:

$$\begin{aligned} \frac{\varphi(N)}{N} &= \prod_{p_i|N}\left(1-\frac{1}{p_i}\right) \geq \left(1-\frac{1}{p_{\min}}\right)^n \\ &\geq e^{-\frac{2n}{p_{\min}}} \geq e^{-\frac{2}{n^2}} \geq 1-\frac{2}{n^2}. \end{aligned} \tag{3.11}$$

By using Equation (3.11), we obtain:

$$\begin{aligned} \frac{2\pi(N-\varphi(N))}{N} + \frac{\pi^2}{4N^2} &\leq 2\pi\left(1-\frac{\varphi(N)}{N}\right) + \frac{\pi^2}{4N^2} \leq 2\pi\frac{2}{n^2} + \frac{\pi^2}{4N^2} \\ &\leq \frac{4\pi}{n^2} + \frac{\pi^2}{4n^6} \leq \frac{1}{n}. \end{aligned}$$

In proving the above upper bound, we have have used the lower bound $N \geq p_{\min} \geq n^3$, as established in Equation (3.8). We also assumed n to be large enough. Hence the sum of the lengths of the non-simple intervals is at most $\frac{1}{n}$. □

Proof. [Proof of Lemma 3.3] Let $I := \left(\cos\left(t\frac{\pi}{2N}\right), \cos\left((t+2)\frac{\pi}{2N}\right)\right)$, $J := \left(\cos\left((t+2)\frac{\pi}{2N}\right), \cos\left((t+4)\frac{\pi}{2N}\right)\right)$. We use the trigonometric identity $\cos A - \cos B = 2\sin\left(\frac{A+B}{2}\right)\sin\left(\frac{B-A}{2}\right)$ to obtain:

$$\frac{\cos\left(t\frac{\pi}{2N}\right) - \cos\left((t+2)\frac{\pi}{2N}\right)}{\cos\left((t+2)\frac{\pi}{2N}\right) - \cos\left((t+4)\frac{\pi}{2N}\right)} = \frac{\sin\left((t+1)\frac{\pi}{2N}\right)}{\sin\left((t+3)\frac{\pi}{2N}\right)}.$$

We have the following cases:

Case 1. In this case, we assume $(t+3)\frac{\pi}{2N} \leq \frac{\pi}{2}$. Now we use the well known inequality:

$$\frac{2}{\pi}x \leq \sin(x) \leq x \text{ for } 0 < x \leq \frac{\pi}{2}.$$

This implies that:

$$\frac{2}{\pi}\frac{t+1}{t+3} \leq \frac{\sin\left((t+1)\frac{\pi}{2N}\right)}{\sin\left((t+3)\frac{\pi}{2N}\right)} \leq 1.$$

Since $\frac{t+1}{t+3} \geq \frac{1}{2}$. We obtain that:

$$\frac{1}{\pi} \leq \frac{\sin\left((t+1)\frac{\pi}{2N}\right)}{\sin\left((t+3)\frac{\pi}{2N}\right)} \leq 1.$$

Case 2. Now consider the case when $(t+3)\frac{\pi}{2N} > \frac{\pi}{2}$. By using the equality $\sin(\pi-\theta) = \sin(\theta)$, we obtain:

$$\frac{\sin\left((t+1)\frac{\pi}{2N}\right)}{\sin\left((t+3)\frac{\pi}{2N}\right)} = \frac{\sin\left((2N-(t+1))\frac{\pi}{2N}\right)}{\sin\left((2N-(t+3))\frac{\pi}{2N}\right)}.$$

In this case, we also know that $(2N-(t+1))\frac{\pi}{2N} \leq \frac{\pi}{2}$. By using the result of the first case, we know that:

$$\frac{1}{\pi} \leq \frac{\sin\left((2N-(t+3))\frac{\pi}{2N}\right)}{\sin\left((2N-(t+1))\frac{\pi}{2N}\right)} \leq 1,$$

which proves the claim.

□

4 Hardness of Counting Real Roots

We first recall some standard definitions of counting complexity classes from [AB09]. A function $f : \{0,1\}^* \to \mathbb{N}$ is in #P if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine M such that for every $x \in \{0,1\}^*$

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

A function $f : \{0,1\}^* \to \mathbb{N}$ is #P-hard if every $g \in$ #P can be computed in polynomial time, allowing oracle calls to f. We say that f is #P-complete if it is in #P and #P-hard. We denote by #3SAT the problem of computing, for a given 3SAT formula W, the number of satisfying assignments for W. It is well known that #3SAT is #P-complete.

The main result of this section can be restated as:

Theorem 4.1. CountRealRoots *is* #P*-hard.*

As in Section 2.2, for a given 3SAT formula W defined over n literals, we choose n distinct odd primes $p_1, \ldots, p_n$ associated with the literals, and we set $M := \prod_{i\in[n]} p_i$. According to Theorem 2.1, for given W, one can compute in polynomial time an SLP, which computes a univariate integral polynomial $P_M(W)$, which has the same set of real roots as $\mathrm{PolySAT}_M(W)$, albeit with multiplicity two. From Equation (2.7), we obtain the following equality for the number of real roots.

$$\frac{1}{2} Z_{\mathbb{R}}(P_M(W)) = Z_{\mathbb{R}}(\mathrm{PolySAT}_M(W)) = \sum_{\psi \text{ satisfies } W} \varphi(M/\alpha(\psi)). \tag{4.12}$$

Lemma 4.1. *Suppose q is an odd prime. Let $p_1, \ldots, p_n$ be n distinct primes in the arithmetic progression $\{aq + 2 \mid a \in \mathbb{N}_{>0}\}$. As above, define $M := \prod_{i\in[n]} p_i$. Then for any 3SAT formula W defined over n literals, we have:*

$$\#W \bmod q \equiv \frac{1}{2} Z_{\mathbb{R}}(P_M(W)) \bmod q.$$

Proof. By assumption, p_i is an odd prime and $p_i - 1 \equiv 1 \bmod q$. Let $N = M/\alpha(\psi)$ be as in Equation (4.12), say $N = \prod_{i\in I} p_i$ for $I \subseteq [n]$. Then,

$$\varphi(N) \bmod q \equiv \prod_{i\in I}(p_i - 1) \equiv 1 \bmod q.$$

Therefore, using Equation (4.12),

$$Z_{\mathbb{R}}(\mathrm{PolySAT}_M(W)) \equiv \sum_{\psi \text{ satisfies } W} 1 \bmod q \equiv \#W \bmod q,$$

which shows the assertion. □

We also need the following lemma.

Lemma 4.2. *There is a polynomial time algorithm, which on input a natural number n (encoded in unary) computes distinct odd primes $q_1, \ldots, q_n$ and, for each $i \in [n]$, computes a collection $p_{i1}, \ldots, p_{in}$ of distinct primes such that $p_{ij} \equiv 2 \bmod q_i$ for all $i \in [n], j \in [n]$.*

Proof. We can directly use the algorithm developed for Theorem 4.11 of [vzGKS96]. Step 3 of that algorithm can be skipped. Moreover, after replacing the randomized primality testing by deterministic primality testing of [AKS04], the algorithm becomes deterministic polynomial time. □

Proof. [Proof of Theorem 1.3] We reduce #3SAT to CountRealRoots. For given n, we first compute in polynomial time the primes q_i and p_{ij} for $i, j \in [n]$ as in Lemma 4.2. Then we set $M_i := p_{i1}p_{i2}\ldots p_{in}$ for $i \in [n]$. For a given a 3SAT formula W over n literals, we first compute in polynomial time SLPs computing the polynomials $P_{M_i}(W)$ for $i \in [n]$ according to Theorem 2.1. Then we compute the number of real zeros $Z_{\mathbb{R}}(P_{M_i}(W))$ for $i \in [n]$ by oracle calls to CountRealRoots. We divide these even numbers by two and have thus computed $\#W \bmod q_i$ for $i \in [n]$ according to Lemma 4.1. Since $\#W \le 2^n$ and $\prod q_i > 2^{n+1}$, by using efficient algorithms for Chinese remaindering [vzGG13], we can recover $\#W$ in polynomial time. We have thus shown that CountRealRoots is #P-hard. □

REMARK 4.1. *It is only natural to ponder whether the problem* CountRealRoots *is* $\#\mathsf{P}$*-complete. Generally speaking, problems involving the counting of real roots have not yet been established as belonging to* $\#\mathsf{P}$*. For a fascinating discussion on related problems, see [BC04, Section 6].*

5 Complexity of Radicals

5.1 Complexity of Factors For a polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$, we define $L(f)$ as the size of the smallest arithmetic circuit computing f from the variables x_i and *any constants* in the field $\mathbb{F}$.

The *Factor Conjecture* [Bür00, Bür04] is a central question in algebraic complexity theory. It asks whether complexity of factors g of f can be bounded by a polynomial in terms of $L(f)$ and the degree of g.

CONJECTURE 5.1. (FACTOR CONJECTURE) *Over a field* $\mathbb{F}$ *of characteristic zero, we have* $L(g) \le \mathsf{poly}(L(f) + \deg(g))$ *for any factor* g *of* $f \in \mathbb{F}[x_1, \dots, x_n]$*.*

Evidence for this conjecture comes from [Bür04], where it was shown that the conjecture is true for approximate complexity. In addition, the factor conjecture is supported by the following result. It tells us that the only situation in which Conjecture 5.1 could fail is when the factor g occurs with exponentially large multiplicity e.

THEOREM 5.1. ([KAL87, BÜR00]) *If* $f, g, h \in \mathbb{F}[x_1, \dots, x_n]$ *are such that* $f = g^e h$ *with* g, h *being coprime, the multiplicity* e *a positive integer, and* $\mathrm{char}(\mathbb{F}) = 0$*, then, setting* $d = \deg g$*, we have*

$$L(g) = O(M(d)^2 M(de)(L(f) + n + d \log e)).$$

Here $M(m) = O(m^2)$ *stands for the number of arithmetic operations sufficient to multiply two degree* m *polynomials over* $\mathbb{F}$*.*

Now consider the following related conjecture proposed in [DSS22, Conjecture 1].

CONJECTURE 5.2. (RADICAL CONJECTURE) *For a nonzero polynomial* $f \in \mathbb{Z}[x_1, \dots, x_n]$ *we have*

$$\min\{\deg(\mathrm{rad}(f)), L(\mathrm{rad}(f))\} \le \mathsf{poly}(L(f)).$$

This essentially states that either the degree of $\mathrm{rad}(f)$ or the complexity of $\mathrm{rad}(f)$ (or both), are polynomially bounded in the complexity of f. For a univariate polynomial $f \in \mathbb{Z}[x]$, Conjecture 5.2 implies that

$$L(\mathrm{rad}(f)) \le \mathsf{poly}(L(f)). \tag{5.13}$$

This follows since $L(g) = O(\deg(g))$ for any $g \in \mathbb{Z}[x]$.

To arrive at the main contribution of our paper (Theorem 1.2), we rely on the following constructive variant of Equation (5.13), formulated for $\tau(f)$ instead of $L(f)$. (Recall that $\tau(f)$ denotes the complexity for constant-free arithmetic circuits.)

CONJECTURE 1.1. (CONSTRUCTIVE UNIVARIATE RADICAL CONJECTURE) *For any polynomial* $f \in \mathbb{Z}[x]$*, we have* $\tau(\mathrm{rad}(f)) \le \mathsf{poly}(\tau(f))$*. Moreover, there is a randomized polynomial time algorithm which, given an SLP of size* s *computing* f*, constructs an SLP for* $\mathrm{rad}(f)$ *of size* $\mathsf{poly}(s)$ *with success probability at least* $1 - \frac{1}{\Omega(s^{1+\epsilon})}$ *for some* $\epsilon > 0$*.*

REMARK 5.1. *The main difference between these radical conjectures, in comparison with the factor conjecture, is that they make a statement about the complexity of factors* g *having huge degree. While there is currently not much evidence for these radical conjectures, one may still view Theorem 5.1 as an indication towards them, since it shows that small multiplicities help. When passing to the radical to* f*, all multiplicities of the irreducible factors* g *of* f *are set to one. As for the plausibility of the constructive version Conjecture 1.1, we just note that statements on the existence of small SLPs usually can be refined to statements concerning randomized effective constructions.*

6 Conclusion and Open Questions

Assuming the constructive radical conjecture (Conjecture 1.1), we proved that PosSLP $\in$ BPP would imply $\mathsf{NP} \subseteq \mathsf{BPP}$. This was achieved by first reducing 3SAT to Unique-SAT [VV86] and then using ideas developed in [Pla84, PS07]. This leads to the first non-trivial lower bound for PosSLP, albeit conditional. Using the ideas developed in [vzGKS96], we also proved that counting the real roots of univariate polynomials computed by a given SLP is #P-hard.

There are several avenues for further research:

1. Of course it remains intriguing to prove the unconditional NP-hardness of PosSLP. In our approach, we first constructed an SLP for the polynomial $P_M(W)$, which has same set of real roots as $\text{PolySAT}_M(W)$, but with multiplicity two. One could try to directly construct a polynomial size SLP for $\text{PolySAT}_M(W)$, which has simple real roots. But this is unlikely because $\text{PolySAT}_M(W) = 1$ iff W is not satisfiable. And we can check $\text{PolySAT}_M(W) = 1$ by polynomial identify testing. This would imply $\mathsf{coNP} \subseteq \mathsf{coRP}$, which is not believed to be true.

2. We saw that the sum-of-square-roots problem and inequality testing of succinctly represented integers are special cases of PosSLP. We do not know of any non-trivial upper or lower bounds for these problems. Perhaps one can first study the complexity of these special cases.

3. In addition, the following special case of PosSLP are worth investigating: A univariate polynomial $f(x) = a + bx^\beta + cx^\gamma \in \mathbb{Z}[x]$ with $\beta, \gamma \in \mathbb{N}$ is called a *trinomial*. Koiran [Koi19] proved that the roots of trinomials are "well-separated" and he posed the following problem: Given a rational p/q and a trinomial $f(x)$ as inputs, determine the sign of $f(p/q)$. This problem is easily seen to be a special case of PosSLP. [BDR22] proved that this problem can be solved in deterministic polynomial time, except on a $\frac{1}{\Omega(\log(dH))}$ fraction of the inputs. Here $\deg(f) \le d$ and $\max\{|a|, |b|, |c|, |p|, |q|\} \le H$. Can we find efficient algorithms for Koiran's problem?

4. [Pla84] proved that the following problem is NP-hard: decide for a given k-sparse polynomial whether it has a root on the unit circle. Can we prove a similar lower bound for *k-sparse polynomials*?

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, USA, 1st edition, 2009.

[ABKPM09] Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. Math. (2)*, 160(2):781–793, 2004.

[BC04] Peter Bürgisser and Felipe Cucker. Variations by complexity theorists on three themes of Euler, Bézout, Betti, and Poincaré. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 73–151. Dept. Math., Seconda Univ. Napoli, Caserta, 2004.

[BCSS97] Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and Real Computation.* Springer-Verlag, Berlin, Heidelberg, 1997.

[BDR22] Eric Boniface, Wei Deng, and J. Maurice Rojas. Trinomials and deterministic complexity limits for real solving, 2022. arXiv:2202.06115.

[BMOR18] Michael A. Bennett, Greg Martin, Kevin O'Bryant, and Andrew Rechnitzer. Explicit bounds for primes in arithmetic progressions. *Illinois Journal of Mathematics*, 62(1-4):427 – 532, 2018.

[Bür00] Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*, volume 7 of *Algorithms and computation in mathematics.* Springer, 2000.

[Bür04] Peter Bürgisser. The complexity of factors of multivariate polynomials. *Found. Comput. Math.*, 4(4):369–396, 2004.

[DSS22] Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. Discovering the roots: Uniform closure results for algebraic classes under factoring. *J. ACM*, 69(3), jun 2022.

[ESY14] Kousha Etessami, Alistair Stewart, and Mihalis Yannakakis. A note on the complexity of comparing succinctly represented integers, with an application to maximum probability parsing. *ACM Trans. Comput. Theory*, 6(2), may 2014.

[GGJ76] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, page 10–22, New York, NY, USA, 1976. Association for Computing Machinery.

[HWHB$^+$08] G.H. Hardy, E.M. Wright, D.R. Heath-Brown, R. Heath-Brown, J. Silverman, and A. Wiles. *An Introduction to the Theory of Numbers*. Oxford mathematics. OUP Oxford, 2008.

[JS12] Gorav Jindal and Thatchaphol Saranurak. Subtraction makes computing integers faster. *CoRR*, abs/1212.2549, 2012.

[Kal87] E. Kaltofen. Single-factor Hensel lifting and its application to the straight-line complexity of certain polynomials. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 443–452, New York, NY, USA, 1987. Association for Computing Machinery.

[KC91] David R. Kincaid and E. Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole, Pacific Grove, Calif., 1991.

[Koi19] Pascal Koiran. Root separation for trinomials. *Journal of Symbolic Computation*, 95:151–161, 2019.

[Kon09] Juha Kontinen. A logical characterization of the counting hierarchy. *ACM Transactions on Computational Logic*, 10, 01 2009.

[Lan13] S. Lang. *Elliptic Curves: Diophantine Analysis*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2013.

[Mal01] Gregorio Malajovich. An effective version of Kronecker's theorem on simultaneous diophantine approximation. Technical report, Technical report, City University of Hong Kong, 2001.

[Pla84] David A. Plaisted. New NP-hard and NP-complete polynomial and integer divisibility problems. *Theoretical Computer Science*, 31(1):125–138, 1984.

[PS07] Daniel Perrucci and Juan Sabia. Real roots of univariate polynomials and straight line programs. *Journal of Discrete Algorithms*, 5(3):471–478, 2007. Selected papers from Ad Hoc Now 2005.

[Sap21] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. `https://github.com/dasarpmar/lowerbounds-survey/releases/download/v9.0.3/fancymain.pdf`, 2021.

[Sud07] Madhu Sudan. 6.841, Lecture Notes: Advanced Complexity Theory. `http://people.seas.harvard.edu/~madhusudan/MIT/ST07/scribe/lect11.pdf`, 2007.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.*, 5(3-4):207–388, 2010.

[VV86] L.G. Valiant and V.V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, third edition, 2013.

[vzGKS96] Joachim von zur Gathen, Marek Karpinski, and Igor Shparlinski. Counting curves and their projections. *computational complexity*, 6(1):64–99, Mar 1996.

[Wag86] Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Inf.*, 23(3):325–356, jun 1986.

[Wal04] Michel Waldschmidt. Open diophantine problems. *Moscow Mathematical Journal*, 4, 01 2004.