On Approximate Computation of Critical Points

Amir Ali Ahmadi* Georgina Hall†

Abstract

We show that computing even very coarse approximations of critical points is intractable for simple classes of nonconvex functions. More concretely, we prove that if there exists a polynomial-time algorithm that takes as input a polynomial in n variables of constant degree (as low as three) and outputs a point whose gradient has Euclidean norm at most 2^n whenever the polynomial has a critical point, then P=NP. The algorithm is permitted to return an arbitrary point when no critical point exists. We also prove hardness results for approximate computation of critical points under additional structural assumptions, including settings in which existence and uniqueness of a critical point are guaranteed, the function is lower bounded, and approximation is measured in terms of distance to a critical point. Overall, our results stand in contrast to the commonly-held belief that, in nonconvex optimization, approximate computation of critical points is a tractable task.

Keywords: critical points, nonconvex optimization, hardness of approximation

1 Introduction

A *critical point* (or a *stationary point*) of a differentiable function is a point at which its gradient vanishes. The problem of computing a critical point is one of the most fundamental algorithmic tasks in mathematical optimization. One key reason for this is that any global (or even local) minimizer of a differentiable function must necessarily be a critical point. Therefore, the task of finding a critical point is often viewed as a prerequisite to that of finding minimizers. Unsurprisingly, structural properties of a function—such as convexity or the Polyak-Łojasiewicz property—that make this task sufficient for finding global minimizers often gain prominence in optimization theory.

Over the years, numerous algorithms have been proposed in the optimization literature to search for critical points under various assumptions; see, e.g., (Bertsekas, 1995; Boyd and Vandenberghe, 2004; Nocedal and Wright, 2006; Cartis et al., 2022) and references therein. Perhaps the two best-known such algorithms are gradient descent, which forms the computational backbone of much of modern machine learning, and Newton's method, which is a key component of interior point methods. Naturally, critical points are fixed points for both of these algorithms. In the case of Newton's method, the aim of each iteration is, in fact, also to find a critical point, namely that of the degree-two Taylor approximation to the original function at the current iterate.

Critical points—and more generally, local notions of optimality—have taken on even greater prominence in the optimization community in recent years, due to the paradigm shift from convex to nonconvex optimization, driven largely by machine learning. Indeed, in the nonconvex setting, it is well understood that global minimization is often an NP-hard problem. A common

*Amir Ali Ahmadi is with the Department of Operations Research and Financial Engineering at Princeton University. Email: aaa@princeton.edu

Amir Ali Ahmadi was partially supported by the Sloan Fellowship, the Princeton AI Lab Seed Grant, the Princeton SEAS Innovation Grant, and a Research Gift in Mathematical Optimization.

†Georgina Hall is with the Decision Sciences Area at INSEAD. Email: georgina.hall@insead.edu

misconception, however, is that this hardness stems primarily from the existence of spurious local minima, i.e., local minima that are not global and that can trap optimization algorithms. This view overlooks the fact that finding even a local minimum is an NP-hard task already for low-degree polynomials (Ahmadi and Zhang, 2022b). The same complexity statement holds for recognizing whether a point is a local minimum (Murty and Kabadi, 1987). Among researchers aware of the hardness results for local optimality, a commonly heard justification for the difficulties encountered is the presence of saddle points or other critical points that hinder algorithmic progress. This is well illustrated, e.g., by a quote from (Ge et al., 2015):

> "The difficulty [of optimizing a nonconvex function] comes from two aspects. First, a nonconvex function may have many local minima and it might be hard to find the best one (global minimum) among them. Second, even finding a local minimum might be hard as there can be many saddle points which have 0-gradient but are not local minima."

A widespread belief reflected in this quote is that finding a critical point should be relatively straightforward——perhaps because critical points are easy to recognize, as the definition requires checking equations at a single point rather than an inequality over a neighborhood (as is the case for local minima), or perhaps because the aforementioned algorithms are expected to converge in a reasonable number of iterations.

Running counter to this common belief, it was recently shown, via a relatively simple polynomial-time reduction from the maximum-cut problem in combinatorial optimization, that deciding if a polynomial of degree 3 or higher has a critical point is NP-hard (Ahmadi and Zhang, 2022a, Theorem 2.1).[1] To reconcile this result with the prior belief, researchers have often made the following assumption: the statement of (Ahmadi and Zhang, 2022a, Theorem 2.1) rules out (unless P=NP) the existence of a polynomial-time algorithm for finding an *exact* critical point, and the difficulty likely stems from the need for exact computation. In practice, however, one is often content with *approximate* computation of a critical point, which is assumed to be a much simpler task. Our goal in this paper is to examine this assumption. As it turns out, our findings go against this intuition.

1.1 Outline, contributions, and related literature

This paper is split into two sections, each covering hardness results relating to a different notion of approximate computation of a critical point of an n-variate differentiable function f. All of our results are in the standard Turing model of computation, where the function f belongs to a specific parametric function class (e.g., low-degree polynomials), which we specify in the formal statements of the theorems. In Section 2, we consider the notion of *ϵ-approximate critical points*, that is, points where the 2-norm of the gradient of f is no more than ϵ. We show in Theorem 3 that unless P=NP, no polynomial-time algorithm can compute an approximate critical point of even extremely poor quality (more precisely, with $\epsilon = 2^n$). Showing this hardness of approximation result is considerably more involved than showing hardness of exact computation of a critical point as done in (Ahmadi and Zhang, 2022a, Theorem 2.1). In a discussion that follows the proof of Theorem 3, we argue that the problem of computing a 2^n-approximate critical point remains intractable for functions that are known to have a unique critical point. In Proposition 10, we show that existence of a polynomial-time algorithm for finding a critical point of a lower bounded function with no spurious critical points (and hence no spurious local minima) implies PPAD=P. Finally, in Theorem 11, we show that even for lower bounded functions that have a critical point in the unit hypercube, unless P=NP, no polynomial-time algorithm can find a 2^n-approximate critical point within a ball of radius 2^n.

[1] The same decision problem for degree-2 polynomials is in P, as it simply requires testing feasibility of a linear system of equations.

In Section 3, we consider the notion of ϵ-*near critical points* of a function f, which are points that are within a distance ϵ from some critical point of f. Our hardness result in Theorem 13 precludes the existence of algorithms that compute ϵ-near critical points in time polynomial in n and $\frac{1}{\epsilon}$. Our final result, Theorem 15, shows that the same statement holds even when the function f is lower bounded. Some concluding remarks are presented in Section 4.

We end our introduction by reviewing related literature on hardness of computation of approximate critical points (i.e., the notion considered in Section 2). We are not aware of any results in the literature regarding hardness of computation of near critical points, discussed in Section 3. In (Carmon et al., 2020), the authors study an oracle-based complexity model in which access to a function is provided through evaluations of the function and its derivatives up to a certain order at queried points. They establish lower bounds on the number of queries required by any algorithm in this model to compute an ϵ-approximate critical point. In (Carmon et al., 2021), the authors derive analogous lower bounds under the additional assumption of convexity. In both of these papers, the hard instances constructed by the authors have a domain whose dimension grows with $\frac{1}{\epsilon}$. In (Hollender and Zampetakis, 2023; Chewi et al., 2023), the authors also provide lower bounds on the number of queries needed in the oracle model to recover an ϵ-approximate critical point, but they do so in fixed and low dimension. The work in (Hollender and Zampetakis, 2023) further presents hardness results for computation of approximate critical points in a model they refer to as the "white box" model. In this model, one is given access to Turing machines which can compute, given an input point x, the evaluation of a function f or of its gradient at x in time polynomial in the size of x. They show that, within this model, in any fixed dimension $n \geq 2$, the problem of finding an approximate critical point is complete for a complexity class known as polynomial local search (PLS), which lies between (the functional versions of) P and NP. In contrast, we do not work in either the oracle, nor the white-box model, but in the Turing model of computation as described above, where the input is the parameters of the function instance from a given (parametric) function class. Our hardness results rule out—unless P=NP—polynomial-time algorithms not only for methods based on function or derivative evaluations (such as gradient descent), but for any computational approach (e.g., those based on convex relaxations followed by rounding procedures). We note that hardness results in fixed dimension, as obtained by (Hollender and Zampetakis, 2023), are not possible in our setting, where the function class of interest is polynomials. Indeed, if the dimension is fixed, even the more general problem of solving polynomial systems of equations and inequalities can be accomplished using quantifier elimination theory and related algorithms (see, e.g., (Grigor'ev and Vorobjov Jr, 1988)) in time polynomial in their degrees and the size of their coefficients. Finally, we remark that there are hardness results on computation of Karush-Kuhn-Tucker points, which are analogs of critical points for constrained optimization problems (Vavasis, 1993; Fearnley et al., 2022, 2025).

2 Complexity of computing an ϵ-approximate critical point

Throughout the paper, we denote the gradient vector of a scalar-valued function f by ∇f.

Definition 1 (Critical point)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. A *critical point* of f is a point $x \in \mathbb{R}^n$ such that $\nabla f(x) = 0$.

As mentioned in the introduction, the paper considers two different notions of approximate computation of a critical point. In this section, we focus on the notion of ϵ-*approximate* critical points, which we define now. Throughout the paper, the notation $||\cdot||$ signifies the 2-norm.

Definition 2 (ϵ-approximate critical point)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function and $\epsilon > 0$. An ϵ-*approximate critical point* of f is a point $x \in \mathbb{R}^n$ such that $||\nabla f(x)|| \leq \epsilon$.

In this section, we present two results regarding the complexity of computing ϵ-approximate critical points. As mentioned in the introduction, all of these results and subsequent ones are

formulated in the standard Turing model of computation. In this model, every instance of the problem has an input representable with a finite number of bits (see, e.g., (Sipser, 1996)). Therefore, we consider differentiable functions that are finitely parametrized. Most of our main results involve polynomials of fixed degree (Theorems 3, 11, 13). Some other results (Theorem 15 and Propositions 9, 10) involve a mixture of polynomials of fixed degree and exponentials of polynomials of fixed degree. The input to our problem instances then consists of the coefficients of the monomials that build up these functions and are taken to be rational numbers. We recall that a *polynomial-time* algorithm is an algorithm whose running time is bounded by a polynomial in the size of the input; i.e., the number of bits required for a binary encoding of all these rational numbers. We further recall that a *pseudo-polynomial-time* algorithm is an algorithm whose running time is bounded by a polynomial in the numeric value of the input.

2.1 Complexity of computing an ϵ-approximate critical point of a general function

We first state the main result of this section. Some of its implications are discussed after the proof.

Theorem 3. *If there is an algorithm $\mathcal{A}$ that takes as input a cubic polynomial p in n variables, runs in polynomial time, and is such that*

(i) when p has no critical points, $\mathcal{A}$ is free to return an arbitrary point,

(ii) when p has a critical point, $\mathcal{A}$ returns a 2^n-approximate critical point,

then $P = NP$.

To prove this theorem, we give a polynomial-time reduction from the following decision problem known as CLIQUE: Given a graph G on m vertices and a positive integer $k \leq m$, decide whether G has a clique of size k, i.e., a set of k pairwise adjacent vertices. It is well known that CLIQUE is NP-complete (Karp, 1975), which implies in particular that, unless P=NP, this problem does not admit a polynomial-time algorithm.

Before presenting this reduction, we prove a handful of technical lemmas. These (bar one) involve showing properties of a quartic polynomial $g : \mathbb{R}^m \to \mathbb{R}$, whose roots are in a one-to-one mapping with cliques in a graph (as formalized in Lemma 4). This polynomial plays a crucial role in the the proof of Theorem 3 as well as in the results of Sections 2.2 and 3. We give its definition first. Let G be a graph with m vertices and adjacency matrix A and let $x = (x_1, \ldots, x_m)$. For constants $k \in \{1, \ldots, m\}$, $C \in \mathbb{R}$, and $N = 160m^2 \cdot (m^2 + 1)$, define

$$g_{G,C,k}(x) := C^2 \left(\left(\sum_{i=1}^{m} x_i - 2k + m \right)^2 + N^2 \sum_{i=1}^{m} (x_i^2 - 1)^2 + \sum_i \sum_{j>i} (1 - A_{ij})^2 (x_i + 1)^2 (x_j + 1)^2 \right). \tag{1}$$

To simplify notation, we omit the dependence on G, C, k and write simply g in our proofs.

Lemma 4. *A graph G on m vertices and with edge set E has a clique of size k if and only if the system of equations*

$$\begin{cases} \sum_{i=1}^{m} x_i = 2k - m \\ x_i^2 - 1 = 0, \ \ i = 1, \ldots, m \\ (x_i + 1)(x_j + 1) = 0, (i,j) \notin E \end{cases} \tag{2}$$

is feasible, or equivalently, if and only if the polynomial $g = g_{G,C,k}$ in (1) vanishes at some point.

The proof of this lemma is left to the reader who can check that the only possible solutions to (2) are indicator vectors of cliques of size k in G (that is, vectors in $\{\pm 1\}^m$ with exactly k entries equal to $+1$, whose indices correspond to vertices of G forming a clique of size k).

In the remainder of the paper, we let $sgn(x)$, for $x \in \mathbb{R}^m$, denote the vector in $\mathbb{R}^m$ whose i-th entry equals 1 if $x_i \geq 0$ and -1 if $x_i < 0$. Our next lemma bounds the value of the polynomial g near the corners of the unit hypercube.

Lemma 5. *Let $\epsilon > 0$, G be a graph on m vertices, $k \in \{1, \ldots, m\}$, and $C \in \mathbb{R}$. Recall the definition of $g = g_{G,C,k}$ in (1) and suppose that $\bar{x} \in \mathbb{R}^m$ satisfies $|\bar{x}_i^2 - 1| \leq \epsilon$ for $i = 1, \ldots, m$. Then,*

$$g(\bar{x}) \geq g(sgn(\bar{x})) - 48C^2m^2 \cdot \epsilon.$$

Proof. Let $\bar{x} \in \mathbb{R}^m$ be such that $|\bar{x}_i^2 - 1| \leq \epsilon$ for $i = 1, \ldots, m$. Fix $i \in \{1, \ldots, m\}$. Then, $1 - \epsilon \leq (\bar{x}_i)^2 \leq 1 + \epsilon$. Since $0 \leq 1 - \epsilon \leq 1$ and $1 \leq 1 + \epsilon \leq 2$, we have

$$1 - \epsilon \leq \sqrt{1 - \epsilon} \leq |\bar{x}_i| \leq \sqrt{1 + \epsilon} \leq 1 + \epsilon. \tag{3}$$

If $sgn(\bar{x}_i) > 0$, then $|\bar{x}_i| = \bar{x}_i$ and (3) implies $sgn(\bar{x}_i) - \epsilon \leq \bar{x}_i \leq sgn(\bar{x}_i) + \epsilon$. Likewise if $sgn(\bar{x}_i) < 0$, then $|\bar{x}_i| = -\bar{x}_i$ and (3) implies $sgn(\bar{x}_i) + \epsilon \geq \bar{x}_i \geq sgn(\bar{x}_i) - \epsilon$. Thus, $|\bar{x}_i - sgn(\bar{x}_i)| \leq \epsilon$. Letting $\epsilon_i = \bar{x}_i - sgn(\bar{x}_i)$, we write $\bar{x}_i = sgn(\bar{x}_i) + \epsilon_i$ with $|\epsilon_i| \leq \epsilon$. Using the fact that $N^2(\bar{x}_i^2 - 1)^2 \geq 0$ for all $i = 1, \ldots, m$, we have

$$g(\bar{x}) \geq C^2 \left(\left(\sum_{i=1}^m \bar{x}_i - 2k + m \right)^2 + \sum_i \sum_{j>i} (1 - A_{ij})^2 (\bar{x}_i + 1)^2 (\bar{x}_j + 1)^2 \right). \tag{4}$$

We now consider each term separately in the above sum replacing $\bar{x}_i = sgn(\bar{x}_i) + \epsilon_i$. We can write

$$\begin{aligned}
&\left(\sum_{i=1}^m \bar{x}_i - 2k + m \right)^2 = \left(\sum_{i=1}^m sgn(\bar{x}_i) + \sum_{i=1}^m \epsilon_i - 2k + m \right)^2 \\
&\geq \left(\sum_{i=1}^m sgn(\bar{x}_i) - 2k + m \right)^2 - 2 \left| \sum_{i=1}^m \epsilon_i \right| \cdot \left| \sum_{i=1}^m sgn(\bar{x}_i) - 2k + m \right| \\
&\geq \left(\sum_{i=1}^m sgn(\bar{x}_i) - 2k + m \right)^2 - 8m^2\epsilon,
\end{aligned} \tag{5}$$

where we have used the fact that $(a+b)^2 \geq a^2 - 2|a| \cdot |b|$ for the first inequality, and the triangle inequality combined to $|\epsilon_i| \leq \epsilon$ and $k \leq m$ for the second. Likewise, we have

$$\begin{aligned}
&(\bar{x}_i + 1)^2 (\bar{x}_j + 1)^2 = ((sgn(\bar{x}_i) + \epsilon_i + 1)(sgn(\bar{x}_j) + \epsilon_j + 1))^2 \\
&\geq (sgn(\bar{x}_i) + 1)^2 (sgn(\bar{x}_j) + 1)^2 \\
&- 2|sgn(\bar{x}_i) + 1| \cdot |sgn(\bar{x}_j) + 1| \cdot |\epsilon_i(sgn(\bar{x}_j) + 1) + \epsilon_j(sgn(\bar{x}_i) + 1) + \epsilon_i\epsilon_j| \\
&\geq (sgn(\bar{x}_i) + 1)^2 (sgn(\bar{x}_j) + 1)^2 - 40\epsilon,
\end{aligned}$$

where we have used the fact that $((a+b)(c+d))^2 = (ac+ad+bc+bd)^2 \geq a^2c^2 - 2|a| \cdot |c| \cdot |ad+bc+bd|$ for the first inequality, and the triangle inequality together with the facts that $|\epsilon_i| \leq \epsilon \leq 1$, $|\epsilon_i\epsilon_j| \leq \epsilon^2 \leq \epsilon$, and $|sgn(x_i^*) + 1| \leq 2$ for any i, j for the second. This implies that

$$\sum_i \sum_{j>i} (1 - A_{ij})^2 (\bar{x}_i + 1)^2 (\bar{x}_j + 1)^2 \geq \sum_i \sum_{j>i} (1 - A_{ij})^2 (sgn(\bar{x}_i) + 1)^2 (sgn(\bar{x}_j) + 1)^2 - 40m^2 \cdot \epsilon.$$

Combining this inequality with (5) in (4), we get our result, i.e.,

$$g(\bar{x}) \geq C^2 \left(\left(\sum_{i=1}^{m} sgn(\bar{x}_i) - 2k + m \right)^2 + \sum_i \sum_{j>i} (1 - A_{ij})^2 (sgn(\bar{x}_i) + 1)^2 (sgn(\bar{x}_j) + 1)^2 - 48m^2 \cdot \epsilon \right)$$
$$= g(sgn(\bar{x})) - 48C^2 m^2 \cdot \epsilon.$$

■

Our next lemma bounds the value of the polynomial g on corners of the unit hypercube that do not correspond to cliques of a desired size.

Lemma 6. *Let G be a graph on m vertices, $k \in \{1, \ldots, m\}$, and $C \in \mathbb{R}$. Recall the definition of $g = g_{G,C,k}$ in (1). For any $s \in \{\pm 1\}^m$ such that the support set $\{i \in \{1, \ldots, m\} \mid s_i = 1\}$ does not correspond to a clique of size k in G, we have*

$$g(s) \geq \frac{C^2}{(m^2+1)^2}.$$

Proof. Let $s \in \{\pm 1\}^m$ be such that the set $\{i \in \{1, \ldots, m\} \mid s_i = 1\}$ does not correspond to a clique of size k in G. It must be the case that

$$\left| \sum_{i=1}^{m} s_i - 2k + m \right| + \sum_i \sum_{j>i} |(1 - A_{ij})(s_i + 1)(s_j + 1)| \geq 1. \tag{6}$$

Indeed, when $s \in \{-1, 1\}^m$, the previous expression only takes integer values and, as a consequence of Lemma 4, it cannot be zero. As there are at most $m^2 + 1$ terms in the sum on the left hand side of (6), it must be the case that either $|\sum_{i=1}^{m} s_i - 2k + m| \geq \frac{1}{m^2+1}$, or $(1 - A_{ij})|(s_i + 1)(s_j + 1)| \geq \frac{1}{m^2+1}$ for some (i, j) with $j > i$. In other words,

$$\left(\sum_{i=1}^{m} s_i - 2k + m \right)^2 \geq \frac{1}{(m^2+1)^2} \text{ or } (1 - A_{ij})^2 (s_i + 1)^2 (s_j + 1)^2 \geq \frac{1}{(m^2+1)^2} \text{ for } (i, j),\ j > i. \tag{7}$$

From (1), we get the conclusion. ■

A final lemma, which does not involve the polynomial g, is needed to prove Theorem 3. It is derived from a bound of (Fujiwara, 1916) on the magnitude of roots of univariate polynomials.

Lemma 7. *Consider the univariate function $f(t) = -\alpha_3 |t|^3 + \alpha_2 |t|^2 + \alpha_1 |t| + \alpha_0$ for $t \in \mathbb{R}$, where $\alpha_0, \alpha_1, \alpha_2, \alpha_3 > 0$. If $t^* \in \mathbb{R}$ is such that $f(t^*) \geq 0$, then*

$$|t^*| \leq 2 \max \left\{ \frac{\alpha_2}{\alpha_3}, \sqrt{\frac{\alpha_1}{\alpha_3}}, \sqrt[3]{\frac{\alpha_0}{2\alpha_3}} \right\}.$$

Proof. Let

$$u := 2 \max \left\{ \frac{\alpha_2}{\alpha_3}, \sqrt{\frac{\alpha_1}{\alpha_3}}, \sqrt[3]{\frac{\alpha_0}{2\alpha_3}} \right\} > 0.$$

Suppose there exists $t^* \in \mathbb{R}$ with $f(t^*) \geq 0$ and $|t^*| > u$, i.e., $t^* > u$ or $t^* < -u$. Without loss of generality, we take $t^* > u$; otherwise, we simply consider $-t^*$ as $f(t^*) = f(-t^*)$. Let $\bar{f}(t) = -\alpha_3 t^3 + \alpha_2 t^2 + \alpha_1 t + \alpha_0$. From the intermediate value theorem, as $\bar{f}$ is continuous with $\bar{f}(t^*) = f(t^*) \geq 0$ and $\lim_{t \to \infty} \bar{f}(t) = -\infty$, it must be the case that $\bar{f}$ has a root $\bar{t}$ such that $\bar{t} \geq t^* > u$. This contradicts Fujiwara's bound (Fujiwara, 1916). ■

We are now ready to prove Theorem 3. This requires the introduction, on top of $x \in \mathbb{R}^m$, of two new vectors of variables:

$$y := (y_0, \dots, y_m) \in \mathbb{R}^{m+1} \text{ and } z := (z_{12}, \dots, z_{1m}, z_{23}, \dots z_{(m-1)m}) \in \mathbb{R}^{m(m-1)/2}.$$

Proof of Theorem 3. We prove that such an algorithm would solve CLIQUE in polynomial time. Let G be a graph on m vertices with adjacency matrix A and let $k \in \{1, \dots, m\}$. Let $n := m + (m+1) + \frac{m(m-1)}{2}$, $N = 160m^2 \cdot (m^2+1)$, and set $C = 2^{n+1}(m^2+1)$. Define p to be the n-variate cubic polynomial

$$p(x,y,z) := C\left(y_0\left(\sum_{i=1}^m x_i - 2k + m\right) + N \cdot \sum_{i=1}^m y_i\left(x_i^2 - 1\right) + \sum_i \sum_{j>i} z_{ij}(1 - A_{ij})(x_i+1)(x_j+1)\right). \tag{8}$$

The reduction from (G,k) to p is polynomial in length. Indeed, all coefficients of p are of the form $C \cdot O(n^2)$, thus requiring $\lceil \log_2(C) \rceil + \lceil \log_2(O(n^2)) \rceil = O(n)$ bits to write down. Furthermore, p has $O(n^2)$ such coefficients. The gradient of p is

$$\nabla p(x,y,z) = \begin{pmatrix} \nabla_x p(x,y,z) \\ \nabla_y p(x,y,z) \\ \nabla_z p(x,y,z) \end{pmatrix},$$

where

$$\begin{aligned} \nabla_x p(x,y,z) &= C\begin{pmatrix} y_0 + 2Ny_1x_1 + \sum_{j>1} z_{1j}(1-A_{1j})(x_j+1) \\ \vdots \\ y_0 + 2Ny_mx_m + \sum_{j>m} z_{mj}(1-A_{mj})(x_j+1) \end{pmatrix}, \\ \nabla_y p(x,y,z) &= C\begin{pmatrix} \sum_{i=1}^m x_i - 2k + m \\ N(x_1^2 - 1) \\ \vdots \\ N(x_m^2 - 1) \end{pmatrix}, \end{aligned} \tag{9}$$

and finally, $\nabla_z p(x,y,z)$ has entries $C(1-A_{ij})(x_i+1)(x_j+1)$ for $(i,j) \in \{1,\dots,m\}^2, i<j$.

We prove that if there is a clique of size k in G, then p has a critical point, and if there is no clique of size k in G, then p does not even have a 2^n-approximate critical point. Once we have shown this claim, the theorem follows. To see this, let $(\bar{x}, \bar{y}, \bar{z}) \in \mathbb{R}^n$ be the point returned by algorithm $\mathcal{A}$. If $||\nabla p(\bar{x}, \bar{y}, \bar{z})|| \leq 2^n$, then there is a clique of size k in G per the claim. If $||\nabla p(\bar{x}, \bar{y}, \bar{z})|| > 2^n$, then p has no critical point (otherwise the algorithm would have returned a 2^n-approximate critical point). Per the claim, there is no clique of size k in G. Thus, if an algorithm like $\mathcal{A}$ existed, we would be able to solve CLIQUE in polynomial time, which would imply P=NP.

We first observe that if there is clique of size k in G, then p has a critical point. Indeed, from Lemma 4, if there is a clique of size k in G, (2) is feasible. Letting $\bar{x}$ be a solution to (2), $\bar{y} = 0$, $\bar{z} = 0$, we have $\nabla p(\bar{x}, \bar{y}, \bar{z}) = 0$.

Conversely, suppose that there is no clique of size k in G. We show that in this case,

$$||\nabla p(x,y,z)|| > 2^n, \forall x, y, z. \tag{10}$$

Let $x, y, z \in \mathbb{R}^n$. We have

$$||\nabla p(x,y,z)||^2 \geq ||\nabla_y p(x,y,z)||^2 + ||\nabla_z p(x,y,z)||^2 = g(x),$$

where $g(x)$ is the polynomial defined in (1). Note that g is a quartic polynomial with highest-degree term $C^2\left(N^2 \sum_{i=1}^m x_i^4 + \sum_i \sum_{j>i}(1-A_{ij})^2 x_i^2 x_j^2\right)$, which is positive definite, i.e., positive

for nonzero x. This implies that g is coercive (Jeyakumar et al., 2014, Section 4.2) and thus attains its infimum (Bertsekas, 1995, Appendix A.2). Let x^* be a minimizer of g. We provide a lower bound on $g(x^*)$, which is a lower bound on $g(x)$, which in turn is a lower bound on $||\nabla p(x,y,z)||_2^2$. To do this, we show through a series of claims that $|(x_i^*)^2 - 1| \leq \epsilon$, for some $\epsilon > 0$ and for all $i = 1, \ldots, m$. This enables us to leverage Lemmas 5 and 6 to conclude.

The proof of our claims makes use of the First Order Necessary Condition (FONC) and Second Order Necessary Condition (SONC) for unconstrained minimization (Bertsekas, 1995, Proposition 1.1.1). We thus compute the first and second-order derivatives of g. We have, for $i = 1, \ldots, m$,

$$\frac{\partial g(x)}{\partial x_i} = C^2 \left(2 \left(\sum_{j=1}^{m} x_j - 2k + m \right) + 4N^2 x_i (x_i^2 - 1) + 2 \sum_{j>i} (1 - A_{ij})^2 (x_i + 1)(x_j + 1)^2 \right). \tag{11}$$

Furthermore, again for $i = 1, \ldots, m$, we have

$$\frac{\partial^2 g(x)}{\partial x_i^2} = C^2 \left(2 + 4N^2 (3x_i^2 - 1) + 2 \sum_{j>i} (1 - A_{ij})^2 (x_j + 1)^2 \right). \tag{12}$$

Claim 1: $0.3 \leq |x_i^*| \leq 2.1$ **for** $i = 1, \ldots, m$**.**

We first show the upper bound. As x^* is a minimizer of g, the FONC applied to g at x^* gives us $\nabla g(x^*) = 0$, that is, for any $i = 1, \ldots, m$,

$$-4C^2 N^2 x_i^* ((x_i^*)^2 - 1) = C^2 \left(2 \left(\sum_{j=1}^{m} x_j^* - 2k + m \right) + 2 \sum_{j>i} (1 - A_{ij})^2 (x_i^* + 1)(x_j^* + 1)^2 \right).$$

Taking absolute values on both sides and upper bounding the right hand side of the equality via the triangle inequality, we obtain:

$$4C^2 N^2 |x_i^*| \cdot |(x_i^*)^2 - 1| \leq C^2 \left(2 \sum_{j=1}^{m} |x_j^*| + 4k + 2m + 2 \sum_{j>i} (1 - A_{ij})^2 (|x_i^*| + 1)(|x_j^*| + 1)^2 \right). \tag{13}$$

We can then lower bound the left hand side of the inequality using the reverse triangle inequality. For any $i = 1, \ldots, m$, we obtain

$$4C^2 N^2 |x_i^*| \cdot (|x_i^*|^2 - 1) \leq C^2 \left(2 \sum_{j=1}^{m} |x_j^*| + 4k + 2m + 2 \sum_{j>i} (1 - A_{ij})^2 (|x_i^*| + 1)(|x_j^*| + 1)^2 \right).$$

Let $i_+ \in \arg\max_{i=1,\ldots,m} |x_i^*|$. Taking $i = i_+$ in the previous inequality and noting that, for any $j = 1, \ldots, m$, $|x_j^*| \leq |x_{i+}^*|$, we get:

$$4C^2 N^2 |x_{i_+}^*| \cdot (|x_{i_+}^*|^2 - 1) \leq C^2 \left(2m |x_{i_+}^*| + 6m + 2m (|x_{i_+}^*| + 1)^3 \right),$$

where we have also used the fact that $k \leq m$ and $(1 - A_{ij})^2 \leq 1$. Expanding and rearranging, we get

$$C^2 \left((2m - 4N^2) |x_{i_+}^*|^3 + 6m |x_{i_+}^*|^2 + (8m + 4N^2) |x_{i_+}^*| + 8m \right) \geq 0.$$

Recall that $N = 80m^2 \cdot 2(m^2+1)$ and thus $4N^2 = 25600m^4 \cdot 4(m^2+1)^2$. For any $m \geq 1$, it follows that $C^2(2m - 4N^2) < 0$, $6mC^2 > 0$, $C^2(8m + 4N^2) > 0$, and $8mC^2 > 0$, and Lemma 7 applies. One can easily show that, for any $m \geq 1$,

$$\max\left\{\frac{6m}{4N^2-2m}, \sqrt{\frac{8m+4N^2}{4N^2-2m}}, \sqrt[3]{\frac{8m}{4N^2-2m}}\right\} = \sqrt{\frac{8m+4N^2}{4N^2-2m}}.$$

Lemma 7 then implies that

$$|x^*_{i_+}| \leq 2\sqrt{\frac{8m+4N^2}{4N^2-2m}} = 2\sqrt{1+\frac{10m}{4N^2-2m}} = 2\sqrt{1+\frac{10}{25600m^3\cdot 4(m^2+1)^2-2}} \text{ for } i = 1,\ldots,m.$$

We have that $25600m^3 \cdot 4(m^2+1)^2 - 2 \geq 25600$ for any $m \geq 1$, and thus,

$$|x_i| \leq |x^*_{i_+}| \leq 2 \cdot \sqrt{1+\frac{10}{25600}} \leq 2.1 \text{ for } i = 1,\ldots,m. \tag{14}$$

To obtain a lower bound on $|x^*_i|$, we use the SONC applied to g at x^*. This gives us that $\nabla^2 g(x^*)$ is positive semidefinite and hence has nonnegative diagonals. Recalling (12), for any $i = 1,\ldots,m$, we have

$$C^2\left(2 + 4N^2(3(x^*_i)^2 - 1) + 2\sum_{j>i}(1-A_{ij})^2(x^*_j+1)^2\right) \geq 0,$$

which by the triangle inequality and (14) implies that

$$C^2\left(2 + 4N^2(3(x^*_i)^2 - 1) + 2m(2.1+1)^2\right) \geq 0.$$

Rearranging, we get that

$$(x^*_i)^2 \geq \frac{1}{3} - \frac{2+2m(2.1+1)^2}{12N^2} \geq \frac{1}{3} - \frac{22m}{3\cdot 25600m^4\cdot 4(m^2+1)^2} \geq \frac{1}{3} - \frac{22}{3\cdot 25600} \geq 0.3,$$

where we have used the fact that $2+2m(2.1+1)^2 \leq 22m$ for any $m \geq 1$ for the second inequality, and the fact that $m^3 \cdot 4(m^2+1)^2 \geq 1$ for all $m \geq 1$ for the third inequality. Taking square roots on either side, we end up with the lower bound in Claim 1.

Claim 2: Let $\epsilon := \frac{0.005}{m^2(m^2+1)^2}$. Then $|(x^*_i)^2 - 1| \leq \epsilon$, for $i = 1,\ldots,m$.

We first use Claim 1 to bound $|(x^*_i)^2 - 1|$ in a more refined fashion. Let $i \in \{1,\ldots,m\}$. We have the following set of inequalities:

$$\begin{aligned}
&4C^2N^2 \cdot 0.3 \cdot |(x^*_i)^2 - 1| \leq 4C^2N^2 \cdot |x^*_i| \cdot |(x^*_i)^2 - 1| \\
&\leq C^2\left(2\sum_{j=1}^{m}|x^*_j| + 4k + 2m + 2\sum_{j>i}(1-A_{ij})^2(|x^*_i|+1)(|x^*_j|+1)^2\right) \\
&\leq C^2\left(2m\cdot 2.1 + 6m + 2m(2.1+1)^3\right),
\end{aligned}$$

where the first inequality makes use of Claim 1, the second inequality is exactly (13), and the third inequality leverages (14) and the fact that $k \leq m$ and $1 - A_{ij} \leq 1$. We then obtain Claim 2 by dividing the inequality on both sides by $4C^2N^2 \cdot 0.3$:

$$|(x^*_i)^2 - 1| \leq \frac{2m\cdot 2.1 + 6m + 2m(2.1+1)^3}{4\cdot 0.3N^2} \leq \frac{235m}{25600m^4\cdot 4(m^2+1)^2} \leq \frac{0.005}{m^3(m^2+1)^2} \leq \epsilon. \tag{15}$$

With Claim 2 established, Lemma 5 implies that $g(x^*) \geq g(sgn(x^*)) - 48C^2m^2\epsilon$. Note that $sgn(x^*) \in \{\pm 1\}^m$ but the set $\{i \in \{1,\ldots,m\} \mid sgn(x^*)_i = 1\}$ cannot be a clique of size k in G (as there are no cliques of size k in G by assumption). Thus, from Lemma 6, we have:

$$g(sgn(x^*)) \geq \frac{C^2}{(m^2+1)^2}.$$

Combining the two inequalities, we get $g(x^*) \geq \frac{C^2}{(m^2+1)^2} - 48C^2m^2\epsilon$. Recalling that

$$||\nabla p(x,y,z)||_2^2 \geq g(x)$$

and x^* is a minimizer of g, we obtain:

$$||\nabla p(x,y,z)||_2^2 \geq g(x) \geq g(x^*) \geq \frac{C^2}{(m^2+1)^2} - 48C^2m^2\epsilon.$$

As $\epsilon = \frac{0.005}{m^2(m^2+1)^2}$, $C = 2^{n+1}(m^2+1)$ and $48 \cdot 0.005 \leq 1/4$, it follows that

$$||\nabla p(x,y,z)||_2^2 \geq \frac{2^{2n+2}(m^2+1)^2}{(m^2+1)^2} - \frac{2^{2n+2}(m^2+1)^2 \cdot 48m^2 \cdot 0.005}{m^2(m^2+1)^2} = 2^{2n+2}(1-\frac{1}{4}) \geq 2^{2n}.$$

Taking square roots on both sides then gives us the desired inequality in (10). ■

We remark that if we set $C = 2^{\text{poly}(n)}(m^2+1)$ in the proof instead of $C = 2^{n+1}(m^2+1)$, it follows that, unless P = NP, computing a $2^{\text{poly}(n)}$-approximate critical point is impossible in polynomial time. We have made a more concrete choice for the constant C, as we believe that the statement of Theorem 3 is already sufficiently negative. More interestingly, by setting $C = n^d(m^2+1)$ for some $d \in \mathbb{N}$ in the proof of Theorem 3, we obtain the following corollary, which states that if the aim is to obtain a less crude n^d-approximate critical point, then we can even rule out pseudo-polynomial-time algorithms (unless P = NP).

Corollary 8. *Let d be an arbitrary positive integer. If there is a pseudo-polynomial-time algorithm $\mathcal{A}$ that takes as input a cubic polynomial p in n variables and is such that (i) when p has no critical points, $\mathcal{A}$ is free to return an abritrary point, (ii) when p has a critical point, $\mathcal{A}$ returns a n^d-approximate critical point, then $P = NP$.*

Discussion around Theorem 3.

As a concrete example of the implications of Theorem 3, consider the gradient descent algorithm. Theorem 3 establishes that, unless P = NP, there are n-variate cubic polynomials for which gradient descent, using any efficiently computable stepsize, requires an exponential number of iterations to return even a 2^n-approximate critical point. In fact, as a consequence of more nuanced results in complexity theory, Theorem 3 further implies that such behavior must occur on a "significant portion" of cubic n-variate polynomials; see (Hemaspaandra and Williams, 2012, Corollary 2.2) for a precise complexity-theoretic statement.

More generally, unless P=NP, Theorem 3 precludes the existence of an algorithm that finds an ϵ-approximate critical point of a function in n variables in time polynomial in $(n, 2^{\frac{1}{\epsilon}})$ (as well as in the natural description size of the function instance). Indeed, if such an algorithm existed, then by setting $\epsilon = 2^n$, the resulting runtime would be polynomial in n and the description size of the instance, contradicting Theorem 3.

We also claim that computing a 2^n-approximate critical point remains hard even in the setting where a critical point of p is guaranteed to exist. Indeed, we can show that if there exists a polynomial-time algorithm which takes as input an n-variate cubic polynomial with a promise that this polynomial has a critical point and returns a 2^n-approximate critical point,

then the PLANTED CLIQUE conjecture (Jerrum, 1992; Kučera, 1995) would be refuted. This well-known conjecture involves taking as input two positive integers, m and $k \in \{1, \ldots, m\}$, and creating an Erdös-Rényi graph on m vertices by connecting each pair of vertices independently with probability $1/2$. A clique of size k is then planted in this graph by randomly choosing a subset of k vertices from the m original ones and adding all edges between them. The conjecture, which is widely believed to be true, states that, unless P=NP, recovering this clique in polynomial time is impossible whenever $k = o(\sqrt{m})$ (and $k \geq (2+\epsilon)\log(m)$ for any constant $\epsilon > 0$) (Chen et al., 2025). We omit the proof of our claim above: it uses the same reduction as the proof of Theorem 3 and then shows that if $(\bar{x}, \bar{y}, \bar{z})$ is a 2^n-approximate critical point of the polynomial p in (8) returned by the algorithm, then $sgn(\bar{x}) \in \{\pm 1\}^m$ is an indicator vector of a clique of size k in G.

Note that in the planted clique setting, with high probability, the graph has a unique clique of size k (Bollobás and Erdös, 1976), which implies that the polynomial p built in the aforementioned reduction has a unique critical point. Thus, the additional promise of uniqueness of the critical point (on top of existence) does not make the task of finding an approximate critical point any easier. In fact, combining our reduction with a seminal result of Valiant and Vazirani (Valiant and Vazirani, 1985), we can show that existence of a polynomial-time algorithm for finding a 2^n-approximate critical point of an n-variate cubic polynomial which is promised to have a unique critical point would imply that any problem in NP can be solved in randomized polynomial time (i.e., NP=RP).

As a final remark on Theorem 3, we briefly discuss its implications for the design of higher-order Newton methods (Nesterov, 2021; Silina and Zhang, 2022; Ahmadi et al., 2024; Cartis et al., 2022). Recall that in each iteration of the classical Newton method, one computes a critical point of the second-order Taylor expansion of the objective function around the current iterate. This computation is tractable, as it reduces to solving a linear system and can be performed in time polynomial in the dimension. Higher-order Newton methods aim to improve the convergence rate and oracle complexity of Newton's method by instead working with Taylor expansions of degree three or higher. While recent work has proposed higher-order methods with tractable per-iteration complexity (Nesterov, 2021; Ahmadi et al., 2024), these algorithms are more involved than a direct extension of Newton's method—namely, one that computes (approximately) a critical point of a higher-order Taylor expansion. Theorem 3 shows that, unless P=NP, this most natural higher-order extension of Newton's method lacks tractable per-iteration complexity already when the Taylor expansion has degree three.

2.2 Complexity of computing an ϵ-approximate critical point of a lower bounded function

In this section, we investigate whether computing an approximate critical point of a function remains intractable if the function is known to be lower bounded. Let us observe first that it is NP-hard to decide whether a lower bounded function has a critical point.

Proposition 9. *If there is a polynomial-time algorithm that can decide whether the exponential of a cubic polynomial (which is always lower bounded) has a critical point, then P=NP.*

Proof. A function $f : \mathbb{R}^n \to \mathbb{R}$ has a critical point if and only if e^f, which is lower bounded, has a critical point. Indeed, $\nabla e^{f(x)} = e^{f(x)} \nabla f(x)$. It is known that deciding whether a cubic polynomial has a critical point is NP-hard (Ahmadi and Zhang, 2022a), and this concludes the proof. ■

The next proposition shows that even under additional assumptions, namely guaranteed existence of a critical point and absence of any spurious critical points[2], the problem of com-

[2]We say that a function has no spurious critical points if any critical point is a global minimum. Observe that absence of spurious critical points implies absence of spurious local minima.

puting a critical point of a lower bounded function remains hard. Indeed, for $x \in \mathbb{R}^n$, let $\{\tilde{q}_i(x) = 0, i = 1, \ldots, r\}$ be a set of quadratic equations which is known to have a (rational) solution, but for which finding a solution is intractable. As an example, such a set of quadratic equations can encode the PLANTED CLIQUE problem (see Section 2.1) or the problem of finding a Nash equilibrium in a bimatrix game[3] (see (Daskalakis et al., 2009; Chen and Deng, 2006) for results on hardness of computing a Nash equilibrium and the definition of the complexity class PPAD).

Proposition 10. *If there is a polynomial-time algorithm that takes as input a lower bounded function (the product of an n-variate quartic polynomial and an exponential function in one variable), which is guaranteed to have critical points but no spurious critical points, and returns a critical point, then the* PLANTED CLIQUE *conjecture is refuted and PPAD=P.*

Proof. As outlined above, let $\{\tilde{q}_i(x) = 0, i = 1, \ldots, r\}$ be a set of quadratic equations in n variables whose solution(s) correspond to the planted clique in the setting of the PLANTED CLIQUE problem, or to Nash equilibria of a bimatrix game. Define the function $f : \mathbb{R}^{n+1} \to \mathbb{R}$ as

$$f(x, w) = e^w \cdot \left(\sum_{i=1}^{r} \tilde{q}_i^2(x) \right).$$

Clearly, f is lower bounded by zero. Furthermore, we have

$$\nabla f(x, w) = \begin{pmatrix} 2e^w \cdot \sum_{i=1}^{r} \tilde{q}_i(x) \nabla q_i(x) \\ e^w (\sum_{i=1}^{r} \tilde{q}_i^2(x)) \end{pmatrix}.$$

Thus, any critical point $(\bar{x}, \bar{w})$ of f must be such that $\tilde{q}_i(\bar{x}) = 0$ for $i = 1, \ldots, r$. This implies that at any critical point $(\bar{x}, \bar{w})$, we have $f(\bar{x}, \bar{w}) = 0$, and thus f has no spurious critical points. It also implies that any polynomial-time algorithm returning a critical point $(\bar{x}, \bar{w})$ of f would provide us, through the vector $\bar{x}$, the planted clique of interest or a Nash equilibrium of the game in polynomial time. This establishes the two claims. ■

We remark that Proposition 10 goes against the somewhat commonly-held belief that minimizing a function with no spurious local minima is tractable. We now study the complexity of finding an *approximate* critical point for a lower bounded function. It is known that a lower bounded function always has an ϵ-approximate critical point for any $\epsilon > 0$ (Ekeland, 1974). The question, then, is whether finding such a point is tractable. We now show that finding an approximate critical point within a reasonably-sized set is intractable.

Theorem 11. *If there is an algorithm $\mathcal{A}$ that takes as input a lower bounded quartic polynomial p in n variables, runs in polynomial time, and is such that*

(i) when p does not have a critical point in the hypercube $[-1, 1]^n$, then $\mathcal{A}$ is free to return an arbitrary point,

(ii) when p has a critical point in the hypercube $[-1, 1]^n$, then $\mathcal{A}$ returns a 2^n-approximate critical point in the ball of radius 2^n centered at the origin,

then P=NP.

Note that the guarantees required of $\mathcal{A}$ are very weak: except in the case where a critical point exists in the hypercube $[-1, 1]^n$, the algorithm is permitted to behave arbitrarily, and even in that case, it is only required to return a very coarse approximate critical point in a

[3] A Nash equilibrium can be defined by a set of quadratic inequalities which ensure no unilateral payoff improvement when moving to a pure strategy. One can then turn this into a set of quadratic equations by introducing slack variables.

very loose ball. Our sense is that, in most applications, one cares about finding ϵ-approximate critical points which are not of unreasonably large size. Such settings are already covered by Theorem 11. We leave open the complexity of finding approximate critical points of lower bounded functions if one is content with the norm of the algorithm output being exponentially large in the dimension.

Proof of Theorem 11. We prove that such an algorithm would solve CLIQUE in polynomial time. Let G be a graph on m vertices with adjacency matrix A and let $k \in \{1, \ldots, m\}$. Let $n = m + (m+1) + \frac{m(m-1)}{2}$, $N = 80m^2 \cdot 2(m^2+1)$, and set $C = 2^{n+2}(m^2+1)$. Define $q(x)$ to be the vector of size $m + 1 + \frac{m(m-1)}{2}$ with entries $q_i(x)$ given by

$$q_1(x) = \sum_{i=1}^{m} x_i - 2k + m, \quad q_2(x) = N(x_1^2 - 1), \quad \ldots, \quad q_{m+1}(x) = N(x_m^2 - 1),$$
$$q_{m+2}(x) = (1 - A_{12})(x_1 + 1)(x_2 + 1), \quad \ldots, \quad q_{m+1+m(m-1)/2} = (1 - A_{(m-1)m})(x_{m-1} + 1)(x_m + 1).$$

Letting $w := (w_1, \ldots, w_{m+1+m(m-1)/2})$, we then define p to be the n-variate quartic polynomial

$$p(x, w) = \frac{1}{2}||w + Cq(x)||^2.$$

Obviously, $p(x, w)$ is lower bounded by zero. Furthermore, the reduction from (G, k) to p is polynomial in length as the number of bits required to write down the coefficients of p is polynomial in the size of (G, k).

Let B be the ball of radius 2^n centered at the origin and let $\mathcal{H} = [-1, 1]^n$. We prove that (i') if there is a clique of size k in G, then p has a critical point in $\mathcal{H}$, and (ii') if there is no clique of size k in G, then p does not even have a 2^n-approximate critical point within B. Once we have shown claims (i') and (ii'), the theorem follows. To see this, let $(\bar{x}, \bar{w})$ be the point returned by the algorithm $\mathcal{A}$. If $||(\bar{x}, \bar{w})|| \leq 2^n$ and $||\nabla p(\bar{x}, \bar{w})|| \leq 2^n$, then G has a clique of size k per claim (ii'). If $||(\bar{x}, \bar{w})|| > 2^n$ or $||\nabla p(\bar{x}, \bar{w}))|| > 2^n$, then p does not have a critical point in $\mathcal{H}$ (otherwise the algorithm would have returned a 2^n-approximate critical point in $\mathcal{B}$) and there is no clique of size k in G, per claim (i'). Thus, if an algorithm such as $\mathcal{A}$ existed, we would be able to solve CLIQUE in polynomial time, which would imply P=NP.

We show claim (i') first; i.e., if there is a clique of size k in G, then p has a critical point within $\mathcal{H}$. To see this, we use Lemma 4 once again to conclude that (2) is feasible. Noting that

$$\nabla p(x, w) = \begin{pmatrix} CJ_q(x)^T(w + Cq(x)) \\ w + Cq(x) \end{pmatrix}, \tag{16}$$

where $J_q(x)$ is the Jacobian of q, and letting $\bar{w} = 0$ and $\bar{x}$ a solution to (2), we have that $\nabla p(\bar{x}, \bar{w}) = 0$. Furthermore, $\bar{x} \in \{\pm 1\}^m$ and $\bar{w} = 0$, which implies that $(\bar{x}, \bar{w}) \in \mathcal{H}$.

We now show claim (ii'). We prove that if there is no clique of size k in G, then

$$||\nabla p(x, w)|| > 2^n, \forall (x, w) \in B.$$

Let $(x, w) \in B$. Note that $||\nabla p(x, w)||^2 \geq ||w + Cq(x)||^2$. Using the reverse triangle inequality, we obtain

$$||\nabla p(x, w)|| \geq ||w + Cq(x)|| \geq |C||q(x)|| - ||w|| \; |. \tag{17}$$

Note that $C^2||q(x)||^2 = g(x)$ as defined in (1). As in the proof of Theorem 3, as Claim 1 and Claim 2 in that proof hold for a minimizer x^* of g and for $\epsilon = \frac{0.005}{m^2(m^2+1)^2}$, we can utilize Lemma 5 and conclude that $g(x^*) \geq g(sgn(x^*)) - 48C^2m^2\epsilon$. As there is no clique of size k in G by assumption, we can apply Lemma 6 once again to obtain $g(sgn(x^*)) \geq \frac{C^2}{(m^2+1)^2}$. Combining the two inequalities, we get

$$g(x) \geq g(x^*) \geq \frac{C^2}{(m^2+1)^2} - 48C^2m^2\epsilon.$$

Replacing ϵ by its value and taking square roots, it follows that

$$C||q(x)|| = \sqrt{g(x)} \geq \frac{\sqrt{3}C}{2(m^2+1)}.$$

Plugging this into (17) and noting that as $(x, w) \in B$, we have $||w|| \leq 2^n$, we get

$$||\nabla p(x, w)|| \geq \frac{\sqrt{3}C}{2(m^2+1)} - 2^n.$$

Recalling the choice of C, we see that $||\nabla p(x, w)|| > 2^n$. This concludes the proof. ■

3 Complexity of computing an ϵ-near critical point

In this section, we focus on the notion of *ϵ-near critical points*, which we define below.

Definition 12 (ϵ-near critical point)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function and $\epsilon > 0$. An *ϵ-near critical point* of f is a point $\bar{x} \in \mathbb{R}^n$ such that $||\bar{x} - x^*|| \leq \epsilon$ for some critical point x^* of f.

The notion of a near critical point is a very natural alternative to that of an approximate critical point. Indeed, a critical point x^* is an optimal solution to the optimization problem $\min_x ||\nabla f(x)||$. An approximate critical point thus is a point that approximates the *optimal value* of this problem, whereas a near critical point is an approximation of the *optimal solution* of this problem; see Figure 1 for an illustration of the the distinction between both notions in a one-dimensional setting. The notion of near critical points is perhaps not as widely used in the literature as that of approximate critical points. We suspect that this is because recognizing whether a given point is an ϵ-near critical point of a function for a given $\epsilon > 0$ is NP-hard[4], as the following simple reduction from Clique demonstrates. Let G be a graph on m vertices and $k \in \{1, \ldots, m\}$. Let p be the n-variate cubic polynomial defined in (8), $u = 0$ (the origin in $\mathbb{R}^n$), and $\epsilon = \lceil\sqrt{m}\rceil$. We have seen in the proof of Theorem 3 that if G has a clique of size k, then there is a critical point of p with $x^* \in \{\pm 1\}^m, y^* = 0, z^* = 0$ (for which x^* is the indicator vector of this clique). Thus, u is an $\sqrt{m}$-near critical point of p. If G does not have a clique of size k, then p has no critical points and so p has no ϵ'-near critical point for any $\epsilon' > 0$ (in particular for $\epsilon' = \lceil\sqrt{m}\rceil$).

In the next subsections, we study the complexity of computing near critical points.

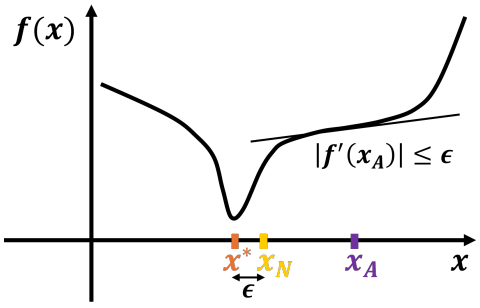

Figure 1: The difference between near critical and approximate critical points demonstrated in a one-dimensional example: x^* is a critical point of f, and for a given $\epsilon > 0$, x_N is an ϵ-near critical point while x_A is an ϵ-approximate critical point.

[4]The same statement is true for the problem of recognizing whether a point $\bar{x}$ is a local minimum (Murty and Kabadi, 1987).

3.1 Complexity of computing an ϵ-near critical point of a general function

Our first result is the analog of Theorem 3 for near critical points.

Theorem 13. *If there is an algorithm $\mathcal{A}$ that takes as input a cubic polynomial p in n variables, runs in polynomial time, and is such that*

(i) when p has no critical point, $\mathcal{A}$ is free to return an arbitrary point,

(ii) when p has a critical point, $\mathcal{A}$ returns a $\frac{1}{12000 \cdot n^{3.5}}$-near critical point,

then $P = NP$.

To prove this theorem, we first show a technical lemma. Recall that a function f is L-Lipschitz over a box $B \subset \mathbb{R}^n$ if $|f(x) - f(y)| \leq L||x - y||, \forall x, y \in B$.

Lemma 14. *Let G be a graph on m vertices, $k \in \{1, \ldots, m\}$, $C \in \mathbb{R}$, and $N = 160m^2(m^2+1)$. Recall the definition of the polynomial $g = g_{G,C,k}$ in (1). For $\epsilon \in (0, 1)$, g is L-Lipschitz over $[-1-\epsilon, 1+\epsilon]^m$ with $L = 64C^2m^2 + 24C^2mN^2\epsilon$.*

Proof. Let $\epsilon \in (0, 1)$ and $x \in [-1-\epsilon, 1+\epsilon]^m$. It suffices to show that $||\nabla g(x)|| \leq L$. We have

$$||\nabla g(x)|| \leq ||\nabla g(x)||_1 = \sum_{i=1}^{m} \left| \frac{\partial g(x)}{\partial x_i} \right|.$$

Fix $i \in \{1, \ldots, m\}$. From (11), we get

$$\begin{aligned}
\left| \frac{\partial g(x)}{\partial x_i} \right| &= \left| C^2 \left(2 \left(\sum_{j=1}^{m} x_j - 2k + m \right) + 4N^2 x_i (x_i^2 - 1) + 2 \sum_{j>i} (1 - A_{ij})^2 (x_i + 1)(x_j + 1)^2 \right) \right| \\
&\leq C^2 \left(2 \left(\sum_{j=1}^{m} |x_j| + 3m \right) + 4N^2 |x_i| \cdot |x_i^2 - 1| + 2 \sum_{j>i} (|x_i| + 1) \cdot (|x_j| + 1)^2 \right) \\
&\leq C^2 \left(2m(1+\epsilon) + 6m + 4N^2(1+\epsilon)(3\epsilon) + 2m(2+\epsilon)^3 \right) \leq C^2(64m + 4N^2 \cdot 6\epsilon),
\end{aligned}$$

where we have used the triangle inequality and the facts that $k \leq m$ and $(1 - A_{ij}) \leq 1$ in the first inequality, the fact that $|x_j| \leq 1 + \epsilon$ and thus

$$|x_j^2 - 1| = |\ |x_j|^2 - 1| = |\ |x_j| - 1| \cdot |\ |x_j| + 1| \leq \epsilon(2 + \epsilon) \leq 3\epsilon$$

in the second inequality, and the fact that $\epsilon < 1$ in the third. Summing both sides of this inequality over i, we get

$$||\nabla g(x)|| \leq \sum_{i=1}^{m} \left| \frac{\partial g(x)}{\partial x_i} \right| \leq mC^2(64m + 4N^2 \cdot 6\epsilon) = L.$$

■

Proof of Theorem 13. We prove that such an algorithm would solve CLIQUE in polynomial time. Let G be a graph on m vertices and with adjacency matrix A and let $k \in \{1, \ldots, m\}$. We let $n := m + (m+1) + \frac{m(m-1)}{2}$, $N = 160m^2(m^2+1)$, and $C = 1$. Define $p = p(x, y, z)$ to be the n-variate cubic polynomial in (8). As mentioned previously, the reduction from (G, k) to p is polynomial in length.

Let $\alpha = \frac{1}{12000 \cdot n^{3.5}}$. We show that if (x, y, z) is an α-near critical point of p, then $sgn(x)$ is the indicator vector of a clique of size k. Once we have shown this claim, the theorem follows. Indeed, suppose that $(\bar{x}, \bar{y}, \bar{z})$ were the point returned by the algorithm $\mathcal{A}$ in polynomial time.

We can simply check whether $sgn(\bar{x})$ is the indicator function of a clique of size k in G. If it is, G obviously has a clique of size k. If it is not, then it must be the case that $(\bar{x}, \bar{y}, \bar{z})$ is not an α-near critical point per the claim. Thus, p has no critical point (otherwise the algorithm would have returned an α-near critical point). It follows, as shown in the proof of Theorem 3, that there is no clique of size k in G. Thus, if there existed an algorithm such as $\mathcal{A}$, we would be able to solve CLIQUE in polynomial time, implying P=NP.

We now show the claim. Let (x, y, z) be an α-near critical point, that is,

$$\left|\left|\begin{pmatrix} x \\ y \\ z \end{pmatrix} - \begin{pmatrix} x^* \\ y^* \\ z^* \end{pmatrix}\right|\right| \leq \alpha = \frac{1}{12000 \cdot n^{3.5}},$$

for some critical point (x^*, y^*, z^*) of p. (Note that if (x, y, z) is α-near critical for p, it must be the case that a critical point of p exists.) We show that $s := sgn(x)$ is the indicator vector of a clique of size k in G. Let $g(x)$ be as defined in (1). To prove the claim, we show first that $g(s) < \frac{1}{(m+1)^4}$ and then that $g(z) \geq \frac{1}{(m+1)^4}$ for any $z \in \{-1, 1\}^m$ that is not an indicator vector of a clique of size k in G.

As (x, y, z) is an α-near critical point of p, we have $||x - x^*|| \leq \alpha$, and therefore

$$|x_i - x_i^*| \leq \alpha, \forall i = 1, \ldots, m.$$

Furthermore, $x^* \in \{\pm 1\}^m$ as (x^*, y^*, z^*) is a critical point of p (see (9)). Thus,

$$\begin{aligned} |x_i^2 - 1| &= |x_i^2 - (x_i^*)^2| = |x_i - x_i^*| \cdot |x_i + x_i^*| = |x_i - x_i^*| \cdot |x_i - x_i^* + 2x_i^*| \\ &\leq |x_i - x_i^*| \cdot (|x_i - x_i^*| + 2|x_i^*|) = |x_i - x_i^*|^2 + 2|x_i - x_i^*| \leq \alpha^2 + 2\alpha, \text{ for } i = 1, \ldots, m. \end{aligned}$$

Applying Lemma 5 and recalling that $C = 1$, we conclude that

$$g(s) \leq g(x) + 48m^2(\alpha^2 + 2\alpha). \tag{18}$$

We can now upper bound $g(x)$ using Lemma 14. As

$$|x_i - x_i^*| \leq \alpha, \forall i = 1, \ldots, m,$$

and $x^* \in \{\pm 1\}^m$, we have that $x, x^* \in [-1 - \alpha, 1 + \alpha]^m$. As $\alpha < 1$, Lemma 14 implies that

$$|g(x) - g(x^*)| = |g(x)| \leq (64m^2 + 24mN^2\alpha)||x - x^*|| \leq (64m^2 + 24mN^2\alpha)\alpha,$$

where we have used the fact that $g(x^*) = 0$ as $\nabla_y p(x^*, y^*, z^*) = 0$. Plugging this back into (18) and substituting the values of N and α, we get:

$$\begin{aligned} g(s) &\leq (64m^2 + 24mN^2\alpha) \cdot \alpha + 48m^2(\alpha^2 + 2\alpha) \\ &= \frac{64m^2}{12000n^{3.5}} + \frac{24m \cdot 160^2 m^4 (m^2+1)^2}{12000^2 n^7} + 48m^2 \left(\frac{1}{12000^2 n^7} + \frac{2}{12000n^{3.5}}\right) \\ &\leq \frac{64m^2 2^{3.5}}{12000(m+1)^7} + \frac{24 \cdot 160^2 m^5 (m^2+1)^2 \cdot 2^7}{12000^2 (m+1)^{14}} + 48m^2 \left(\frac{2^7}{12000^2 (m+1)^{14}} + \frac{2 \cdot 2^{3.5}}{12000(m+1)^7}\right) \\ &\leq \frac{0.1}{(m+1)^5} + \frac{0.6}{(m+1)^5} + \frac{0.01}{(m+1)^{12}} + \frac{0.1}{(m+1)^5} < \frac{1}{(m+1)^4}, \end{aligned}$$

where we have used the inequality $n \geq \frac{(m+1)^2}{2}$ in the third line and $(m^2 + 1) \leq (m + 1)^2$ in the fourth.

Now, let $z \in \{\pm 1\}^m$ and suppose that z is not an indicator vector of a clique of size k in G. We have

$$g(z) \geq \frac{C^2}{(m^2+1)^2} = \frac{1}{(m^2+1)^2} \geq \frac{1}{(m+1)^4},$$

from Lemma 6, where we have used the fact that $m^2 + 1 \leq (m + 1)^2$. This concludes the proof. ■

Discussion around Theorem 13.

Unless P=NP, Theorem 13 precludes the existence of an algorithm that finds an ϵ-near critical point of a function in n variables in time polynomial in $(n, \frac{1}{\epsilon})$ (as well as in the natural description size of the function instance). Indeed, if such an algorithm existed, then by setting $\epsilon = \frac{1}{12000n^{3.5}}$, the resulting algorithm would be polynomial in n and the description size of the instance, contradicting Theorem 13. Note that since the size of ϵ is $\log(1/\epsilon)$, the above statement in fact rules out algorithms that have exponential running time in the size of ϵ.

3.2 Complexity of computing an ϵ-near critical point of a lower bounded function

In a parallel to Section 2.2, we show that computing an ϵ-near critical point of a function f remains hard under the additional assumption that f is lower bounded.

Theorem 15. *If there is an algorithm $\mathcal{A}$ that takes as input a lower bounded function f, which is the product of an n-variate quartic polynomial and an exponential function in one variable, runs in polynomial time, and is such that*

(i) when f has no critical point, $\mathcal{A}$ is free to return an abitrary point,

(ii) when f has a critical point, $\mathcal{A}$ returns a $\frac{1}{1000 \cdot n^7}$-near critical point,

then P=NP.

It is worth remarking that despite the factors in Theorems 13 and 15 being different, the conclusion in the discussion around Theorem 13 (see above) applies identically to Theorem 15, i.e., to the lower bounded case.

Proof of Theorem 15. We prove that such an algorithm would solve Clique in polynomial time. Let G be a graph on m vertices and with adjacency matrix A and let $k \in \{1, \ldots, m\}$. Let $n = m + 1$, $N = 160m^2(m^2 + 1)$, and $C = 1$. Define the n-variate function

$$f(x, w) = e^w \cdot g(x)$$

in variables $(x, w) \in \mathbb{R}^m \times \mathbb{R}$, where $g(x)$ is as given in (1). Clearly, f is lower bounded by zero and, as mentioned in the proof of Proposition 10, the reduction from (G, k) to f is polynomial in length. The gradient of f is given by

$$\nabla f(x, w) = \begin{pmatrix} e^w \nabla g(x) \\ e^w g(x) \end{pmatrix}.$$

Let $\alpha = \frac{1}{1000n^7}$. Similarly to the proof of Theorem 13, we show the following claim: if (x, w) is an α-near critical point of f, then $sgn(x)$ is the indicator vector of a clique of size k. The theorem follows in an identical fashion to what is described in the proof of Theorem 13.

We show the claim now. Let (x, w) be an α-near critical point of f, i.e.,

$$||(x, w) - (x^*, w^*)|| \leq \alpha,$$

where (x^*, w^*) is a critical point of f. Note that any critical point (x^*, w^*) of f must be such that $g(x^*) = 0$. To prove the claim, we show first that $g(sgn(x)) < \frac{1}{(m+1)^4}$ and then that $g(z) \geq \frac{1}{(m+1)^4}$ for any $z \in \{-1, 1\}^m$ which is not an indicator vector of a clique of size k in G.

Reprising identical arguments to the proof of Theorem 13, it can be shown that

$$|x_i^2 - 1| \leq \alpha^2 + 2\alpha, \forall i = 1, \ldots, m.$$

Using Lemma 5 and Lemma 14, we obtain analogously

$$g(sgn(x)) \leq g(x) + 48m^2(\alpha^2 + 2\alpha) \leq L\alpha + 48m^2(\alpha^2 + 2\alpha),$$

where $L = 64C^2m^2 + 24C^2mN^2\alpha$. Substituting in the values of α, C, and n, we get

$$\begin{aligned} g(sgn(x)) &\leq \frac{64m^2}{1000(m+1)^7} + \frac{24m \cdot 160^2 m^4 (m^2+1)^2}{1000^2(m+1)^{14}} + 48m^2 \left(\frac{1}{1000^2(m+1)^{14}} + \frac{2}{1000(m+1)^7} \right) \\ &\leq \frac{0.1}{(m+1)^5} + \frac{0.65}{(m+1)^5} + \frac{0.01}{(m+1)^{12}} + \frac{0.1}{(m+1)^5} < \frac{1}{(m+1)^4}, \end{aligned}$$

where we have used the fact that $m^2 + 1 \leq (m+1)^2$.

Now let $z \in \{-1, 1\}^m$ and suppose that z is not an indicator vector of a clique of size k in G. From Lemma 6, it follows that $g(z) \geq \frac{1}{(m^2+1)^2} \geq \frac{1}{(m+1)^4}$. This completes the proof. ■

4 Concluding remarks

We have shown that computing even poor-quality approximations of critical points is intractable for simple classes of nonconvex functions, and that this remains true when the existence and uniqueness of a critical point are guaranteed or when the function is lower bounded. It is possible that future work could make our hardness results even more negative, for example, by utilizing hardness of approximation results for Clique (Håstad, 1996). On the positive side, we believe our results motivate research in the following direction: What are classes of parametric functions and structural assumptions on these functions such that (i) ϵ-approximate or ϵ-near critical points can be computed for this function class in time polynomial in $\log(1/\epsilon)$ and the size of the parameters, and (ii) the structural property on the function can be checked in time polynomial in the size of the parameters? For example, a recent result (Slot et al., 2025) implies that an ϵ-approximate and ϵ-near critical point for convex polynomials of a fixed degree can be computed in time polynomial in $\log(1/\epsilon)$ and the size of the coefficients of the polynomial, in line with (i). However, the structural property of convexity does not satisfy (ii) for polynomials of degree 4 or higher (Ahmadi et al., 2013), though there are sufficient conditions for convexity that do (see, e.g., (Ahmadi and Hall, 2018, Definition 5)). What are other function classes and structural properties that make approximate computation of critical points tractable in the formal sense that we have specified above?

References

Amir Ali Ahmadi and Georgina Hall. DC decomposition of nonconvex polynomials with algebraic techniques. *Mathematical Programming*, 169(1):69–94, 2018.

Amir Ali Ahmadi and Jeffrey Zhang. Complexity aspects of local minima and related notions. *Advances in Mathematics*, 397:108119, 2022a.

Amir Ali Ahmadi and Jeffrey Zhang. On the complexity of finding a local minimizer of a quadratic function over a polytope. *Mathematical Programming*, 195(1):783–792, 2022b.

Amir Ali Ahmadi, Alex Olshevsky, Pablo A Parrilo, and John N Tsitsiklis. NP-hardness of deciding convexity of quartic polynomials and related problems. *Mathematical Programming*, 137(1):453–476, 2013.

Amir Ali Ahmadi, Abraar Chaudhry, and Jeffrey Zhang. Higher-order Newton methods with polynomial work per iteration. *Advances in Mathematics*, 452:109808, 2024.

Dimitri P Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, 1995.

Béla Bollobás and Paul Erdös. Cliques in random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 80, pages 419–427. Cambridge University Press, 1976.

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points I. *Mathematical Programming*, 184(1):71–120, 2020.

Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points II: First-order methods. *Mathematical Programming*, 185(1):315–355, 2021.

Coralia Cartis, Nicholas IM Gould, and Philippe L Toint. *Evaluation Complexity of Algorithms for Nonconvex Optimization: Theory, Computation and Perspectives*. SIAM, 2022.

Xi Chen and Xiaotie Deng. Settling the complexity of two-player Nash equilibrium. In *Foundations of Computer Science*, volume 6, pages 261–272, 2006.

Zongchen Chen, Elchanan Mossel, and Ilias Zadik. Almost-linear planted cliques elude the Metropolis process. *Random Structures & Algorithms*, 66(2):e21274, 2025.

Sinho Chewi, Sébastien Bubeck, and Adil Salim. On the complexity of finding stationary points of smooth functions in one dimension. In *International Conference on Algorithmic Learning Theory*, pages 358–374. PMLR, 2023.

Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a Nash equilibrium. *Communications of the ACM*, 52(2):89–97, 2009.

Ivar Ekeland. On the variational principle. *Journal of Mathematical Analysis and Applications*, 47(2):324–353, 1974.

John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS= PPAD ∩ PLS. *Journal of the ACM*, 70(1):1–74, 2022.

John Fearnley, Paul Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of computing KKT solutions of quadratic programs. *Journal of the ACM*, 72(5):1–48, 2025.

Matsusaburô Fujiwara. Über die obere Schranke des absoluten Betrages der Wurzeln einer algebraischen Gleichung. *Tohoku Mathematical Journal, First Series*, 10:167–171, 1916.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pages 797–842. PMLR, 2015.

D Yu Grigor'ev and Nicolai N Vorobjov Jr. Solving systems of polynomial inequalities in subexponential time. *Journal of Symbolic Computation*, 5(1-2):37–64, 1988.

Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. In *Foundations of Computer Science*, pages 627–636, 1996.

Lane A Hemaspaandra and Ryan Williams. An atypical survey of typical-case heuristic algorithms. *ACM SIGACT News*, 43(4):70–89, 2012.

Alexandros Hollender and Emmanouil Zampetakis. The computational complexity of finding stationary points in non-convex optimization. In *Conference on Learning Theory*, pages 5571–5572. PMLR, 2023.

Mark Jerrum. Large cliques elude the Metropolis process. *Random Structures & Algorithms*, 3 (4):347–359, 1992.

Vaithilingam Jeyakumar, Jean B Lasserre, and Guoyin Li. On polynomial optimization over non-compact semi-algebraic sets. *Journal of Optimization Theory and Applications*, 163: 707–718, 2014.

Richard M Karp. On the computational complexity of combinatorial problems. *Networks*, 5(1): 45–68, 1975.

Luděk Kučera. Expected complexity of graph partitioning problems. *Discrete Applied Mathematics*, 57(2-3):193–212, 1995.

Katta G. Murty and Santosh N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39(2):117–129, 1987.

Yurii Nesterov. Implementable tensor methods in unconstrained convex optimization. *Mathematical Programming*, 186(1):157–183, 2021.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 2006.

Olha Silina and Jeffrey Zhang. An unregularized third order Newton method. *arXiv preprint arXiv:2209.10051*, 2022.

Michael Sipser. Introduction to the Theory of Computation. *ACM Sigact News*, 27(1):27–29, 1996.

Lucas Slot, David Steurer, and Manuel Wiedmer. Hesse's redemption: Efficient convex polynomial programming. *arXiv preprint arXiv:2511.03440*, 2025.

Leslie G Valiant and Vijay V Vazirani. NP is as easy as detecting unique solutions. In *ACM Symposium on Theory of Computing*, pages 458–463, 1985.

Stephen A Vavasis. Black-box complexity of local minimization. *SIAM Journal on Optimization*, 3(1):60–80, 1993.