The Complexity of Constrained Min-Max Optimization

Constantinos Daskalakis
costis@csail.mit.edu
Massachusetts Institute of Technology
USA

Stratis Skoulakis
efstratios@sutd.edu.sg
Singapore University of Technology & Design, Singapore

Manolis Zampetakis
mzampet@berkeley.edu
UC Berkeley
USA

ABSTRACT

Despite its important applications in Machine Learning, min-max optimization of objective functions that are *nonconvex-nonconcave* remains elusive. Not only are there no known first-order methods converging to even approximate local min-max equilibria (a.k.a. approximate saddle points), but the computational complexity of identifying them is also poorly understood. In this paper, we provide a characterization of the computational complexity as well as of the limitations of first-order methods in this problem. Specifically, we show that in linearly constrained min-max optimization problems with nonconvex-nonconcave objectives an approximate local min-max equilibrium of large enough approximation is guaranteed to exist, but computing such a point is PPAD-complete. The same is true of computing an approximate fixed point of the (Projected) Gradient Descent/Ascent update dynamics, which is computationally equivalent to computing approximate local min-max equilibria. An important byproduct of our proof is to establish an unconditional hardness result in the Nemirovsky-Yudin [36] oracle optimization model, where we are given oracle access to the values of some function $f : \mathcal{P} \to [-1, 1]$ and its gradient ∇f, where $\mathcal{P} \subseteq [0, 1]^d$ is a known convex polytope. We show that any algorithm that uses such first-order oracle access to f and finds an ε-approximate local min-max equilibrium needs to make a number of oracle queries that is exponential in at least one of $1/\varepsilon$, L, G, or d, where L and G are respectively the smoothness and Lipschitzness of f. This comes in sharp contrast to minimization problems, where finding approximate local minima in the same setting can be done with Projected Gradient Descent using $O(L/\varepsilon)$ many queries. Our result is the first to show an exponential separation between these two fundamental optimization problems in the oracle model.

CCS CONCEPTS

• **Theory of computation → Problems, reductions and completeness**; Complexity classes; • **Mathematics of computing → Nonconvex optimization**.

KEYWORDS

min-max optimization, computational complexity, black-box lower bounds, TFNP, PPAD

STOC '21, June 21–25, 2021, Virtual, Italy

ACM ISBN 978-1-4503-8053-9/21/06.
https://doi.org/10.1145/3406325.3451125

ACM Reference Format:
Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis. 2021. The Complexity of Constrained Min-Max Optimization. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3406325.3451125

1 INTRODUCTION

Min-Max Optimization has played a central role in the development of Game Theory [43], Convex Optimization [2, 10], and Online Learning [6–8, 21, 41, 42]. In its general constrained form, it can be written down as follows:

$$\min_{\boldsymbol{x}\in\mathbb{R}^{d_1}} \max_{\boldsymbol{y}\in\mathbb{R}^{d_2}} f(\boldsymbol{x},\boldsymbol{y}); \quad \text{s.t. } g(\boldsymbol{x},\boldsymbol{y}) \le 0. \tag{1.1}$$

Here, $f : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to [-B, B]$ with $B \in \mathbb{R}_+$, and $g : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}$ is typically taken to be a convex function so that the constraint set $g(\boldsymbol{x},\boldsymbol{y}) \le 0$ is convex. In this paper, we only use linear functions g so the constraint set is a polytope, thus projecting on this set and checking feasibility of a point with respect to this set can both be done in polynomial time.

The goal in (1.1) is to find a feasible pair $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$, i.e., $g(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \le 0$, that satisfies the following

$$f(\boldsymbol{x}^\star,\boldsymbol{y}^\star) \le f(\boldsymbol{x},\boldsymbol{y}^\star), \quad \text{for all } \boldsymbol{x} \text{ s.t. } g(\boldsymbol{x},\boldsymbol{y}^\star) \le 0; \tag{1.2}$$

$$f(\boldsymbol{x}^\star,\boldsymbol{y}^\star) \ge f(\boldsymbol{x}^\star,\boldsymbol{y}), \quad \text{for all } \boldsymbol{y} \text{ s.t. } g(\boldsymbol{x}^\star,\boldsymbol{y}) \le 0. \tag{1.3}$$

It is well-known that, when $f(\boldsymbol{x},\boldsymbol{y})$ is a convex-concave function, i.e., f is convex in $\boldsymbol{x}$ for all $\boldsymbol{y}$ and it is concave in $\boldsymbol{y}$ for all $\boldsymbol{x}$, then Problem (1.1) is guaranteed to have a solution, under compactness of the constraint set [39, 43], while computing a solution is amenable to convex programming. In fact, if f is L-smooth, the problem can be solved via first-order methods, which are iterative, only access f through its gradient,[1] and achieve an approximation error of $\text{poly}(L, 1/T)$ in T iterations; see e.g. [27, 35].[2] When the function is strongly convex-strongly concave, the rate becomes linear [18].

Unfortunately, our ability to solve Problem (1.1) remains rather poor in settings where our objective function f is *not* convex-concave. This is emerging as a major challenge in Deep Learning, where min-max optimization has recently found many important applications, such as training Generative Adversarial Networks (see e.g. [5, 20]), and robustifying deep neural network-based models against adversarial attacks (see e.g. [29]). These applications are

[1] In general, the access to the constraints g by these methods is more involved, namely through an optimization oracle that optimizes convex functions (in fact, quadratic suffices) over $g(\boldsymbol{x},\boldsymbol{y}) \le 0$. In the settings considered in this paper g is linear and these tasks are computationally straightforward.

[2] In the stated error rate, we are suppressing factors that depend on the diameter of the feasible set. Moreover, the stated error of $\varepsilon(L, T) \triangleq \text{poly}(L, 1/T)$ reflects that these methods return an approximate min-max solution, wherein the inequalities on the LHS of (1.2) and (1.3) are satisfied to within an additive $\varepsilon(L, T)$.

indicative of a broader deep learning paradigm wherein robustness properties of a deep learning system are tested and enforced by another deep learning system. In these applications, it is very common to encounter min-max problems with objectives that are nonconvex-nonconcave, and thus evade treatment by the classical algorithmic toolkit targeting convex-concave objectives.

Indeed, the optimization challenges posed by objectives that are nonconvex-nonconcave are not just theoretical frustration. Practical experience with first-order methods is rife with frustration as well. A common experience is that the training dynamics of first-order methods is unstable, oscillatory or divergent, and the quality of the points encountered in the course of training can be poor; see e.g. [3, 13, 14, 19, 30, 32–34]. This experience is in stark contrast to minimization (resp. maximization) problems, where even for nonconvex (resp. nonconcave) objectives, first-order methods have been found to efficiently converge to approximate local optima or stationary points (see e.g. [4, 23, 28]), while practical methods such Stochastic Gradient Descent, Adagrad, and Adam [16, 26, 38] are driving much of the recent progress in Deep Learning.

The goal of this paper is to *shed light on the complexity of min-max optimization problems*, and *elucidate its difference to minimization and maximization problems*—as far as the latter is concerned without loss of generality we focus on minimization problems, as maximization problems behave exactly the same; we will also think of minimization problems in the framework of (1.1), where the variable $\boldsymbol{y}$ is absent, that is $d_2 = 0$. An important driver of our comparison between min-max optimization and minimization is, of course, the nature of the objective. So let us discuss:

▹ *Convex-Concave Objective.* The benign setting for min-max optimization is that where the objective function is convex-concave, while the benign setting for minimization is that where the objective function is convex. In their corresponding benign settings, the two problems behave quite similarly from a computational perspective in that they are amenable to convex programming, as well as first-order methods which only require gradient information about the objective function. Moreover, in their benign settings, both problems have guaranteed existence of a solution under compactness of the constraint set. Finally, it is clear how to define approximate solutions. We just relax the inequalities on the left side of (1.2) and (1.3) by some $\varepsilon > 0$.

▹ *Nonconvex-Nonconcave Objective.* The challenging setting for min-max optimization is that where the objective is *not* convex-concave, while the challenging setting for minimization is that where the objective is not convex. In these challenging settings, the behavior of the two problems diverges significantly. The first difference is that, while a solution to a minimization problem is still guaranteed to exist under compactness of the constraint set even when the objective is not convex, a solution to a min-max problem is *not* guaranteed to exist when the objective is not convex-concave, even under compactness of the constrained set. A trivial example is this: $\min_{x\in[0,1]} \max_{y\in[0,1]} (x-y)^2$. Unsurprisingly, we show that checking whether a min-max optimization problem has a solution is NP-hard. In fact, we show that checking whether a function $f : [0,1]^d \to [-1,1]$ has an ε-approximate min-max solution is NP-hard, even when the function is G-Lipschitz and L-smooth, and when $1/\varepsilon$, G and L are polynomial in the dimension d.

Since min-max solutions may not exist, what could we plausibly hope to compute? There are two obvious targets:

(I) approximate stationary points of f, as considered e.g. by [1]; and
(II) some type of approximate *local* min-max solution.

Unfortunately, as far as (I) is concerned, it is still possible that (even approximate) stationary points may not exist, and we show that checking if there is one is NP-hard, even when the constraint set is $[0,1]^d$, the objective has Lipschitzness and smoothness polynomial in d, and the desired approximation is an absolute constant (Theorem 4.1). So we focus on (II), i.e. (approximate) local min-max solutions. Several kinds of those have been proposed in the literature [14, 24, 30]. We consider a generalization of the concept of local min-max equilibria, proposed in [14, 30], that also accommodates approximation.

Definition 1.1 (Approximate Local Min-Max Equilibrium)**.** Given f, g as above, and $\varepsilon, \delta > 0$, some point $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is an (ε,δ)-*local min-max solution of* (1.1), or a (ε,δ)-*local min-max equilibrium*, if it is feasible, i.e. $g(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \le 0$, and satisfies:

$$f(\boldsymbol{x}^\star, \boldsymbol{y}^\star) < f(\boldsymbol{x}, \boldsymbol{y}^\star) + \varepsilon, \text{ for all } \boldsymbol{x} \text{ such that } \left\|\boldsymbol{x} - \boldsymbol{x}^\star\right\| \le \delta \text{ and } g(\boldsymbol{x}, \boldsymbol{y}^\star) \le 0; \tag{1.4}$$

$$f(\boldsymbol{x}^\star, \boldsymbol{y}^\star) > f(\boldsymbol{x}^\star, \boldsymbol{y}) - \varepsilon, \text{ for all } \boldsymbol{y} \text{ such that } \left\|\boldsymbol{y} - \boldsymbol{y}^\star\right\| \le \delta \text{ and } g(\boldsymbol{x}^\star, \boldsymbol{y}) \le 0. \tag{1.5}$$

In words, $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is an (ε,δ)-local min-max equilibrium, whenever the min player cannot update $\boldsymbol{x}$ to a feasible point within δ of $\boldsymbol{x}^\star$ to reduce f by at least ε, and symmetrically the max player cannot change $\boldsymbol{y}$ locally to increase f by at least ε.

We show that the existence and complexity of computing such approximate local min-max equilibria depends on the relationship of ε and δ with the smoothness, L, and the Lipschitzness, G, of the objective function f. We distinguish the following regimes, also shown in Figure 1 together with a summary of our associated results.

► **Trivial Regime.** This occurs when $\delta < \frac{\varepsilon}{G}$. This regime is trivial because the G-Lipschitzness of f guarantees that all feasible points are (ε,δ)-local min-max solutions.

► **Local Regime.** This occurs when $\delta < \sqrt{\frac{2\varepsilon}{L}}$, and it represents the interesting regime for min-max optimization. In this regime, we use the smoothness of f to show that (ε,δ)-local min-max solutions always exist. Indeed, we show that computing them is computationally equivalent to the following variant of (I) which is more suitable for the constrained setting:

(I') (approximate) fixed points of the projected gradient descent-ascent dynamics (Section 3.3).

We show via an application of Brouwer's fixed point theorem to the iteration map of the projected gradient descent-ascent dynamics that (I') are guaranteed to exist. In fact, not only do they exist, but computing them is in PPAD, as can be shown by bounding the Lipschitzness of the projected gradient descent-ascent dynamics.

▸ **Global Regime.** This occurs when δ is comparable to the diameter of the constraint set. In this case, the existence of (ε, δ)-local min-max solutions is not guaranteed, and determining their existence is NP-hard, even when the parameters of the problem are bound by a polynomial of the number of dimensions.

The main results of this paper, summarized in Figure 1, are to characterize the complexity of computing local min-max solutions in the local regime. Our first main theorem is the following:

Theorem 1. *Computing (ε, δ)-local min-max solutions of Lipschitz and smooth objectives over convex compact domains in the local regime is* PPAD*-complete. The hardness holds even when the constraint set is a polytope that is a subset of $[0,1]^d$, the objective takes values in $[-1,1]$ and the smoothness, Lipschitzness, $1/\varepsilon$ and $1/\delta$ are polynomial in the dimension. Equivalently, computing α-approximate fixed points of the Projected Gradient Descent-Ascent dynamics on smooth and Lipschitz objectives is* PPAD*-complete, and the hardness holds even when the constraint set is a polytope that is a subset of $[0,1]^d$, the objective takes values in $[-1,1]$ and smoothness, Lipschitzness, and $1/\alpha$ are polynomial in the dimension.*

For the above complexity result we assume that we have "white box" access to the objective function. An important byproduct of our proof, however, is to also establish an *unconditional hardness result* in the Nemirovsky-Yudin [36] oracle optimization model, wherein we are given black-box access to oracles computing the objective function and its gradient. Our second main result is the following.

Theorem 2 (see Theorem 4.5)**.** *Assume that we have black-box access to an oracle computing a G-Lipschitz and L-smooth objective function $f : \mathcal{P} \to [-1,1]$, where $\mathcal{P} \subseteq [0,1]^d$ is a known polytope, and its gradient ∇f. Then, computing an (ε, δ)-local min-max solution in the local regime (i.e., when $\delta < \sqrt{2\varepsilon/L}$) requires a number of oracle queries that is exponential in at least one of the following: $1/\varepsilon$, L, G, or d. In fact, exponential in d-many queries are required even when L, G, $1/\varepsilon$ and $1/\delta$ are all polynomial in d.*

Importantly, the above lower bounds, in both the white-box and the black-box setting, come in sharp contrast to minimization problems, given that finding approximate local minima of smooth non-convex objectives ranging in $[-B, B]$ in the local regime can be done using first-order methods using $O(B \cdot L/\varepsilon)$ time/queries. Our results are the first to show an exponential separation between these two fundamental problems in optimization in the black-box setting, and a super-polynomial separation in the white-box setting assuming PPAD $\neq$ FP.

1.1 Proof Ideas and Overview of Techniques

In this section, we give a brief description of the ideas and techniques that we use to prove our main results, i.e. our PPAD-hardness result and black box lower bound, and refer to the corresponding sections of the paper for complete details. Towards the end of this section, we also comment on the techniques behind our NP-hardness results.

1.1.1 Equivalence of Local Min-Max Equilibria and GDA Fixed Points. The first step of our proof is to show that computing an (ε, δ)-local min-max equilibrium of f in the local regime, namely problem LR-LocalMinMax of Definition 4.2, is computationally equivalent to computing an approximate fixed point of the *projected gradient descent-ascent (GDA) dynamics* on f, namely problem GDAFixed-Point of Section 3.3, which asks for an α-approximate fixed point of the mapping

$$(\boldsymbol{x}, \boldsymbol{y}) \mapsto \left(\Pi_{\mathcal{P}}\left(\boldsymbol{x} - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})\right), \Pi_{\mathcal{P}}\left(\boldsymbol{y} + \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})\right)\right).$$

The computational equivalence between the two problems can be shown using the smoothness properties of f. In light of this equivalence, we show our hardness results for GDAFixedPoint, as this is more convenient for our proofs. In doing so, we establish the hardness of computing approximate fixed points of Projected GDA which is of independent interest. Finally, as GDAFixedPoint is a fixed point computation problem, it is easy to see that it is total (i.e. it always has solutions), and, by showing the Lipschitzness of the Projected GDA map, we also establish that the problem lies in PPAD. By our computational equivalence, the same is true for LR-LocalMinMax.

1.1.2 Proof Sketch of Theorem 1. Given the equivalence discussed in the previous section, to prove Theorem 1 it suffices to prove that it is PPAD-hard to compute α-approximate fixed points of the Projected GDA map for G-Lipschitz and L-smooth functions $f(\boldsymbol{x}, \boldsymbol{y})$ where $(\boldsymbol{x}, \boldsymbol{y})$ is constrained in a polytope and f takes values in $[-1, 1]$. Indeed, we show our hardness result for a simple type of polytope of the form $\mathcal{P} = \{(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x}, \boldsymbol{y} \in [0,1]^d \wedge |x_i - y_i| \leq \Delta, \forall i\}$, and our hardness result will hold even for settings where $G, L, 1/\alpha, 1/\Delta$ are all polynomial in d.

We next provide a sketch of the proof that finding approximate fixed points of Projected GDA is PPAD-hard, the complete proof being provided in the full version of the paper. Our starting point, as in many PPAD-hardness results, is a discrete analog of the problem of finding Brouwer fixed points of a continuous map. Departing from previous work, however, we do not use Sperner's lemma as the discrete analog of Brouwer's fixed point theorem. Instead, we define a new problem, called BiSperner, which is more suitable for our proofs. Similarly to Sperner, BiSperner takes as input a coloring circuit Cl$(\cdot)$ that is used to color vertices $\boldsymbol{v} \in [N]^d$ of a d-dimensional grid. The basic difference between Sperner and BiSperner is that in Sperner there are $d+1$ colors and each vertex is colored with a unique color, whereas in BiSperner there are $2d$ colors and each vertex receives d different colors. More precisely, in the definition of BiSperner we have the following:

- There are $2d$ colors, denoted $1^-, 1^+, \ldots, d^-, d^+$, where we associate the colors j^- and j^+ with coordinate $j \in \{1, \ldots, d\}$.
- For each coordinate $j = 1, \ldots, d$, there exists a mapping Cl$(\cdot, \mathrm{j})$ provided in the input to BiSperner as a Boolean circuit that assigns to each vertex $\boldsymbol{v}$ either color j^+ or j^-. As a result, vertex $\boldsymbol{v}$ is assigned d different colors, i.e., one $+$ or $-$ color for each different coordinate. When Cl$(\boldsymbol{v}, j) = 1$ vertex $\boldsymbol{v}$ receives color j^+, while when Cl$(\boldsymbol{v}, j) = -1$ vertex $\boldsymbol{v}$ receives color j^-.
- Given the above, BiSperner asks for a set of $d+1$ vertices that belong to the same cubelet of the grid $[N]^d$ with the property that every color in $\{1^-, 1^+, \ldots, d^-, d^+\}$ is assigned by Cl to at least one of the $d+1$ vertices.

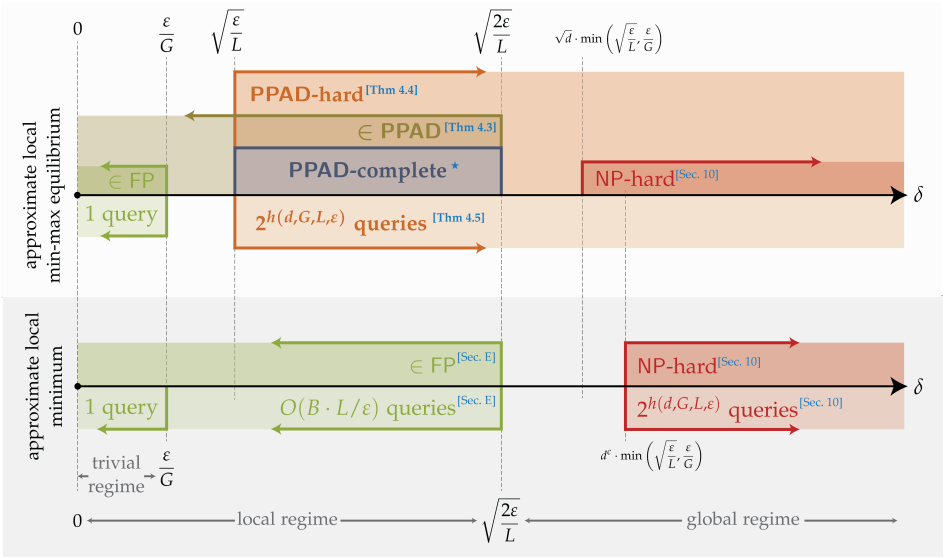


Figure 1: Overview of the results proven in this paper and comparison between the complexity of computing an (ε, δ)-approximate local minimum and an (ε, δ)-approximate local min-max equilibrium of a G-Lipschitz and L-smooth function over a d-dimensional polytope taking values in the interval $[-B, B]$. We assume that $\varepsilon < G^2/L$, thus the trivial regime is a strict subset of the local regime. Moreover, we assume that the approximation parameter ε is provided in unary representation in the input to these problems, which makes our hardness results stronger and the comparison to the upper bounds known for finding approximate local minima fair, as these require time/oracle queries that are polynomial in $1/\varepsilon$. We note that the unary representation is not required for our results proving inclusion in PPAD. The figure portrays a sharp contrast between the computational complexity of approximate local minima and approximate local min-max equilibria in the local regime. Above the black lines, tracking the value of δ, we state our "white box" results and below the black lines we state our "black-box" results. The main result of this paper is the PPAD-hardness of approximate local min-max equilibrium for $\delta \geq \sqrt{\varepsilon/L}$ and the corresponding query lower bound. In the query lower bound the function h is defined as $h(d, G, L, \varepsilon) = \left(\min(d, \sqrt{L/\varepsilon}, G/\varepsilon)\right)^p$ for some universal constant $p \in \mathbb{R}_+$. For the query lower bound of the local-min problem our proof holds for $c = 11$. With $\star$ we indicate our PPAD-completeness result. The tractability of finding approximate local minima is a folklore result. All the section numbers in this figure refer to full-version of our paper [15].

Similar to Sperner, for the existence of a solution to BiSperner the coloring circuit Cl needs to satisfy some boundary conditions. To have a syntactic problem, violations of these conditions are valid outputs for the problem. The boundary coloring conditions are simple: all vertices $\boldsymbol{v}$ of the grid with $v_j = N$ cannot contain color j^+, and all those with $v_j = 0$ cannot contain color j^-.

The first step of our proof is to show that BiSperner is PPAD-hard. This step follows easily via a reduction from finding approximate Brouwer fixed points, which is well-known to be PPAD-complete [9, 37, 40].

The second and main step of our proof is the construction of a function $f(\boldsymbol{x}, \boldsymbol{y})$ such that any approximate fixed point of Projected GDA on f, with constraint set $\mathcal{P}$ as above, corresponds to a solution of a given BiSperner instance with coloring circuit Cl$(\cdot)$. The function $f(\boldsymbol{x}, \boldsymbol{y})$ defines a zero-sum game between two players controlling variables $\boldsymbol{x} \in [0,1]^d$ and $\boldsymbol{y} \in [0,1]^d$ respectively. In order to reduce a given BiSperner instance to the problem of computing an approximate Projected GDA fixed point, we first divide $[0,1]^d$ into N^d cubelets of sidelength $1/(N-1)$, the vertices of which correspond to the vertices of the BiSperner instance's grid. The strategy space of both the minimizing and the maximizing player is in correspondence with the interior of the BiSperner instance's grid. Next, we use the the coloring circuit Cl$(\cdot)$ to define the payoff $f(\boldsymbol{x}, \boldsymbol{y})$ of the zero-sum game, as follows. First, we define a simplicization of the cubelets of $[0,1]^d$ which produces a simplicization of the whole of $[0,1]^d$. Now, for every point $(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]^d \times [0,1]^d$ we map $\boldsymbol{x}$ to the $d+1$ vertices $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d+1}\}$ of the simplex containing $\boldsymbol{x}$. The value of $f(\boldsymbol{x}, \boldsymbol{y})$ is then defined as a function of $\boldsymbol{x}$, $\boldsymbol{y}$ and the coloring sequences that Cl$(\cdot)$ assigns to the vertices in $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d+1}\}$. Before getting into more details about our construction we state that its goal is to satisfy the following property:

Property 1. *For any approximate fixed point $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of Projected GDA on f, constrained in $\mathcal{P}$, strategy $\boldsymbol{x}^*$ must lie in a panchromatic simplex, or a simplex one of whose vertices exhibits a violation of the boundary coloring by* Cl$(\cdot)$.

The main idea behind the construction of the function $f(\boldsymbol{x}, \boldsymbol{y})$ is simple. Specifically, we define

$$f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{d} \alpha_j(\boldsymbol{x}) \cdot (x_j - y_j)$$

where each *interpolation coefficient* $\alpha_j(\boldsymbol{x})$ belongs in $[-1,1]$ and is defined as a function of which of the colors j^+ or j^- the circuit $\mathrm{Cl}(\cdot, j)$ assigns to each of the vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{d+1}$. Despite the fact that the construction of the *interpolation coefficients* is fairly complicated, the key property that we want them to satisfy is the following.

Property 2. *If $\boldsymbol{x}$ lies in a simplex whose vertices are monochromatic with respect to coordinate j (i.e. they either all have color j^- or they all have color j^+), then $\alpha_j(\boldsymbol{x})$ is either -1 or $+1$ (depending on whether $\mathrm{Cl}(\cdot, j)$ assigns color j^- or respectively j^+ to all the vertices of the simplex).*

Using this property we can get intuition about why, when $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an approximate Projected GDA fixed point, the point $\boldsymbol{x}^*$ has to lie in a panchromatic simplex. For the sake of contradiction, let us assume that $\boldsymbol{x}^*$ is contained in a simplex that is monochromatic with respect to some coordinate j, and without loss of generality let us assume that all the vertices of this simplex have color j^+. By our construction, for all $\boldsymbol{x}$ in this simplex, it holds that

$$f(\boldsymbol{x}, \boldsymbol{y}) = (x_j - y_j) + \sum_{\ell \neq j} \alpha_\ell(\boldsymbol{x}) \cdot (x_\ell - y_\ell)\,. \tag{1.6}$$

We construct the coefficients $\alpha_j(\boldsymbol{x})$, and pick Δ in the definition of the feasible set $\mathcal{P}$ to be sufficiently small, so that (1.6) implies that $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial x_j} \cong 1$ and $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial y_j} = -1$. It is easy to see though, that if $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an α-approximate fixed point of Projected GDA that is not near the boundaries of the constraints $|x_j - y_j| \leq \Delta$ and $0 \leq x_j \leq 1$, then both $|\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial x_j}| \leq \alpha < 1$ and $|\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial y_j}| \leq \alpha < 1$, which contradicts our earlier assertion that $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial x_j} \cong 1$ and $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial y_j} = -1$. On the other hand if $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is on the boundary of $|x_j - y_j| \leq \Delta$ then the gradient vector should be almost perpendicular to the boundary of $|x_j - y_j| \leq \Delta$. It is not hard to see that for this to happen the partial derivatives $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial x_j}$ and $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial y_j}$ should be approximately equal and in particular they should have the same sign, which contradicts again our assertion that $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial x_j} \cong 1$ and $\frac{\partial f(\boldsymbol{x}^*, \boldsymbol{y}^*)}{\partial y_j} = -1$. Finally, the more technical cases to analyze are when the monochromatic simplex lies near the boundary of the hypercube, and this is handled in detail in the full version of the paper where we relate this to a violation of the coloring conditions of BiSperner by Cl.

So the only thing that remains to explain is how we define the coefficients $\alpha_j(\boldsymbol{x})$. These coefficients must be appropriately constructed so that each $\alpha_j(\boldsymbol{x})$ is $\mathrm{poly}(N,d)$-Lipschitz and $\mathrm{poly}(N,d)$-smooth. This way we can ensure the Lipschitzness and smoothness of f as well. A traditional way to ensure continuity is through the convex combination produced by the vertices of the *canonical simplicization*. The canonical simplicization is an elegant way to partition $[0,1]^d$ into $d! \cdot N^d$ simplices which also admits a polynomial time algorithm for computing the $d+1$ vertices of a simplex, $\mathrm{Simplex}(\boldsymbol{x})$, of the canonical simplicization in which a point $\boldsymbol{x} \in [0,1]^d$ belongs. Using this classical idea we could attempt to define the coefficients $\alpha(\boldsymbol{x})$ as follows

$$\alpha_j(\boldsymbol{x}) = \sum_{\boldsymbol{v} \in \mathrm{Simplex}(\boldsymbol{x})} \mathrm{Cl}(\boldsymbol{v}, j) \cdot \ell_{\boldsymbol{v}}(\boldsymbol{x}), \tag{1.7}$$

where $\{\ell_{\boldsymbol{v}}(\boldsymbol{x})\}_{\boldsymbol{v} \in \mathrm{Simplex}(\boldsymbol{x})}$ are the unique coefficients such that $\sum_{\boldsymbol{v} \in \mathrm{Simplex}(\boldsymbol{x})} \ell_{\boldsymbol{v}}(\boldsymbol{x}) \boldsymbol{v} = \boldsymbol{x}$. Notice that the above construction ensures that $\alpha_j(\boldsymbol{x})$ is 1, respectively -1, if $\boldsymbol{x}$ belongs to a simplex that is j^+-monochromatic, respectively j^--monochromatic. Moroever, each interpolation coefficient $\alpha_j(\boldsymbol{x})$ is a continuous function of $\boldsymbol{x}$, since whenever $\boldsymbol{x}$ lies in two or more simplices (i.e. lies on a face shared by two or more neighboring simplices), the coefficient $\ell_{\boldsymbol{v}}(\boldsymbol{x}) = 0$ for all vertices $\boldsymbol{v}$ not belonging in that face. Moreover it is not very hard to prove that $\alpha_j(\boldsymbol{x})$ is $\Theta(N \cdot \sqrt{d})$-Lipschitz. The main issue with the above construction is that the coefficients are not *smooth*, indeed $\alpha_j(\boldsymbol{x})$ is not even differentiable.

To tackle this problem we replace the coefficients $\ell_v(\boldsymbol{x})$ with our novel interpolation coefficients that we call *smooth and efficient interpolation coefficients*.

Smooth and Efficient Interpolation Coefficients. In order to establish both continuity and smoothness of the interpolation coefficients $\ell_v(\boldsymbol{x})$, we need to go far beyond the above simple linear convex combination of the vertices of the simplex that contains $\boldsymbol{x}$. The definition of these coefficients is fairly technical and for this reason we defer the details to the full version of the paper, but we summarize here the properties that $\ell_v(\boldsymbol{x})$ should satisfy **i)** they define a convex combination, i.e., $\ell_{\boldsymbol{v}}(\boldsymbol{x}) \geq 0$ and $\sum_{\boldsymbol{v} \in \mathrm{Simplex}(\boldsymbol{x})} \ell_{\boldsymbol{v}}(\boldsymbol{x}) = 1$; this implies that using these coefficients in (1.7) results in Property 2 being satisfied, **ii)** let $\Phi_{\boldsymbol{v}}(\boldsymbol{x})$ be the face of the simplex $\mathrm{Simplex}(\boldsymbol{x})$ that does not contain $\boldsymbol{v}$, then for every $\boldsymbol{z} \in \Phi_{\boldsymbol{v}}(\boldsymbol{x})$ it holds $\ell_{\boldsymbol{v}}(\boldsymbol{z}) = 0$ and $\nabla \ell_{\boldsymbol{v}}(\boldsymbol{z}) = \boldsymbol{0}$, **iii)** all $\ell_v(\boldsymbol{x})$ are $\mathrm{poly}(N,d)$-Lipschitz and $\mathrm{poly}(N,d)$-smooth functions of $\boldsymbol{x}$, and **iv)** $\ell_v(\boldsymbol{x})$ can be computed in polynomial time. Once these properties are satisfied our reduction is independent of the exact construction. We believe though that the construction of the coefficients $\ell_v(\boldsymbol{x})$ is of independent interest with possible future application in proving hardness of continuous problems. In particular, these coefficients can be used as a tool to transfer well known hardness results from discrete problems to hardness of continuous problems. As an example beyond our main result we use our smooth and efficient interpolation coefficients to provide a very simple proof of a lower bound on the number of black-box queries needed in order to compute an (ε, δ)-*local minimum* when δ is large enough.

The definition of the smooth and efficient interpolation coefficients completes our proof sketch of Theorem 1. The formal version of this theorem is presented in the full version of the paper.

1.1.3 Proof Sketch of Theorem 2. We now briefly present a sketch of the proof of our black box result whose exact statement can be found in Theorem 4.5. The proof is based on our proof of Theorem 1 that we presented in the previous section combined with the seminal result of [22] about the black box lower bound on computing Brouwer fixed points. In particular, [22] proves that given *black-box access* to a mapping $M : [0,1]^d \to [0,1]^d$ that is L-Lipschitz continuous, we need at least $\Omega\left((L/\varepsilon)^d\right)$ value queries to M to find an ε-approximate fixed point of M.

More precisely, the first step using the proof of PPAD-hardness of BiSperner we prove that if we only have black box query access to the coloring circuit Cl of BiSperner then $\Omega\left(N^d\right)$ number of queries are necessary to find a solution to BiSperner.

The next step follows from the proof that we presented in Section 1.1.2. In particular, we prove that given an instance Cl of BiSperner there exists a G-Lipschitz and L-smooth function $f : [0,1]^d \times [0,1]^d \to \mathbb{R}$ with the following properties: (1) any (ε, δ)-approximate local min-max equilibria of f can be used to efficiently compute a solution of the initial BiSperner instance, (2) given any point $(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]^d \times [0,1]^d$ it is possible to compute $f(\boldsymbol{x}, \boldsymbol{y})$ and $\nabla f(\boldsymbol{x}, \boldsymbol{y})$ using only $d+1$ black box queries on the initial Cl instance of BiSperner.

If we combine the above two step this implies that it is impossible to exist a black-box algorithm to compute an (ε, δ)-approximate local min-max equilibrium of f that uses less than $\Omega\left(\left(\frac{L}{\varepsilon}\right)^d \frac{1}{d+1}\right)$ queries, since this contradicts the lower bound of [22] for computing approximate Brouwer fixed points. The formal statement and the proof of Theorem 2 is presented in the full version of the paper.

1.1.4 Overview of NP-Hardness Results. Closing this section we mention that all our NP-hardness results are proven using an application of Lovász Local Lemma [17], which provides a powerful rounding tool that can drive the inapproximability to be low.

1.2 Local Minimization vs Local Min-Max Optimization

Because our proof is convoluted, involving multiple steps, it is difficult to discern from it why finding local min-max solutions is so much harder than finding local minima. For this reason, we illustrate in this section a fundamental difference between local minimization and local min-max optimization. This provides good intuition about why our hardness construction would fail if we tried to apply it to prove hardness results for finding local minima (which we know don't exist).

So let us illustrate a key difference between min-max problems that can be expressed in the form $\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y)$, i.e. two-player zero-sum games wherein the players optimize opposing objectives, and min-min problems of the form $\min_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} f(x, y)$, i.e., two-player coordination games wherein the players optimize the same objective. For simplicity, suppose $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ and let us consider long paths of best-response dynamics in the strategy space, $\mathcal{X} \times \mathcal{Y}$, of the two players; these are paths along which at least one of the players improves their payoff. For our illustration, suppose that the derivative of the function with respect to either variable is either 1 or -1. Consider a long path of best-response dynamics starting at a pair of strategies (x_0, y_0) in either a min-min problem or a min-max problem, and a specific point (x, y) along that path. We claim that in min-min problems the function value at (x, y) will have to reveal how far from (x_0, y_0) point (x, y) lies within the path in ℓ_1 distance. On the other hand, in min-max problems the function value at (x, y) may reveal very little about how far (x, y) lies from (x_0, y_0). We illustrate this in Figure 2. While in our min-min example the function value must be monotonically decreasing inside the best-response path, in the min-max example the function values repeat themselves in every straight line segment of length 3, without revealing where in the path each segment is.

Ultimately a key difference between min-min and min-max optimization is that best-response paths in min-max optimization problems can be closed, i.e., can form a cycle, as shown in Figure 2, Panel (b). On the other hand, this is impossible in min-min problems as the function value must monotonically decrease along best-response paths, thus cycles may not exist.

The above discussion offers qualitative differences between min-min and min-max optimization, which lie in the heart of why our computational intractability results are possible to prove for min-max but not min-min problems.

1.3 Further Related Work

There is a broad literature on the complexity of equilibrium computation. Virtually all these results are obtained within the computational complexity formalism of *total search problems* in NP, which was spearheaded by [25, 31, 37] to capture the complexity of search problems that are guaranteed to have a solution. In the full version of the paper, we give a non-exhaustive list of intractability results for equilibrium computation.

1.4 Roadmap of the Paper

In Section 2 and Section 3 we provide notation, the definitions of the computational problems that we consider, and preliminaries. In Section 4 we present the formal statements of our main results together with some complementary results of the paper. All the formal proofs of our results can be found in our full-version of our paper [15].

2 PRELIMINARIES

Notation. For any compact and convex $K \subseteq \mathbb{R}^d$ and $B \in \mathbb{R}_+$, we define $L_\infty(K, B)$ to be the set of all continuous functions $f : K \to \mathbb{R}$ such that $\max_{\boldsymbol{x} \in K} |f(\boldsymbol{x})| \leq B$. When $K = [0,1]^d$, we use $L_\infty(B)$ instead of $L_\infty([0,1]^d, B)$ for ease of notation. For $p > 0$, we define $\mathrm{diam}_p(K) = \max_{\boldsymbol{x}, \boldsymbol{y} \in K} \|\boldsymbol{x} - \boldsymbol{y}\|_p$, where $\|\cdot\|_p$ is the usual ℓ_p-norm of vectors. For an alphabet set Σ, the set Σ^*, called the Kleene star of Σ, is equal to $\cup_{i=0}^{\infty} \Sigma^i$. For any string $\boldsymbol{q} \in \Sigma$ we use $|\boldsymbol{q}|$ to denote the length of $\boldsymbol{q}$. We use the symbol $\log(\cdot)$ for base 2 logarithms and $\ln(\cdot)$ for the natural logarithm. We use $[n] \triangleq \{1, \ldots, n\}$, $[n] - 1 \triangleq \{0, \ldots, n-1\}$, and $[n]_0 \triangleq \{0, \ldots, n\}$.

Lipschitzness, Smoothness, and Normalization. Our main objects of study are continuously differentiable Lipschitz and smooth functions $f : \mathcal{P} \to \mathbb{R}$, where $\mathcal{P} \subseteq [0,1]^d$ is some polytope. A continuously differentiable function f is called *G-Lipschitz* if $|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq G \|\boldsymbol{x} - \boldsymbol{y}\|_2$, for all $\boldsymbol{x}, \boldsymbol{y}$, and *$L$-smooth* if

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L \|\boldsymbol{x} - \boldsymbol{y}\|_2, \quad \text{for all} \boldsymbol{x}, \boldsymbol{y}.$$

Remark 2.1 (Function Normalization). *Note that the G-Lipschitzness of a function $f : \mathcal{P} \to \mathbb{R}$, where $\mathcal{P} \subseteq [0,1]^d$ implies that for any $\boldsymbol{x}$ and $\boldsymbol{y}$ it holds that $|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq G\sqrt{d}$. Whenever the range of a G-Lipschitz function is taken to be $[-B, B]$, for some B, we always assume that $B \leq G\sqrt{d}$. This can be accomplished by setting $\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}) - f(\boldsymbol{x}_0)$ for some fixed $\boldsymbol{x}_0$ in the domain of f. For all the problems that we consider in this paper any solution for $\tilde{f}$ is also a solution for f and vice-versa.*

Function Access. We study optimization problems involving real-valued functions, considering two access models to such functions.

- **Black Box Model.** In this model we are given access to an oracle O_f such that given a point $\boldsymbol{x} \in [0,1]^d$ the oracle O_f returns the values $f(\boldsymbol{x})$ and $\nabla f(\boldsymbol{x})$. In this model we assume that we can perform real number arithmetic operations. This is the traditional model used to prove lower bounds in Optimization and Machine Learning [36].
- **White Box Model.** In this model we are given the description of a polynomial-time Turing machine C_f that computes $f(\boldsymbol{x})$ and $\nabla f(\boldsymbol{x})$. More precisely, given some input $\boldsymbol{x} \in [0,1]^d$, described using B bits, and some accuracy ε, C_f runs in time upper bounded by some polynomial in B and $\log(1/\varepsilon)$ and outputs approximate values for $f(\boldsymbol{x})$ and $\nabla f(\boldsymbol{x})$, with approximation error that is at most ε in ℓ_2 distance. We note that a running time upper bound on a given Turing Machine can be enforced syntactically by stopping the computation and outputting a fixed output whenever the computation exceeds the bound. See also Remark 2.6 for an important remark about how to formally study the computational complexity of problems that take as input a polynomial-time Turing Machine.

Promise Problems. To simplify the exposition of our paper, make the definitions of our computational problems and theorem statements clearer, and make our intractability results stronger, we choose to enforce the following constraints on our function access, O_f or C_f, as a *promise*, rather than enforcing these constraints in some syntactic manner.

(1) **Consistency of Function Values and Gradient Values.** Given some oracle O_f or Turing machine C_f, it is difficult to determine by querying the oracle or examining the description of the Turing machine whether the function and gradient values output on different inputs are consistent with some differentiable function. In all our computational problems, we will only consider instances where this is promised to be the case. Moreover, for all our computational hardness results, the instances of the problems arising from our reductions satisfy these constraints, which are guaranteed syntactically by our reduction.

(2) **Lipschitzness, Smoothness and Boundedness.** Similarly, given some oracle O_f or Turing machine C_f, it is difficult to determine, by querying the oracle or examining the description of the Turing machine, whether the function and gradient values output by O_f or C_f are consistent with some Lipschitz, smooth and bounded function with some prescribed Lipschitzness, smoothness, and bound on its absolute value. In all our computational problems, we only consider instances where the G-Lipschitzness, L-smoothness and B-boundedness of the function are promised to hold for the prescribed, in the input of the problem, parameters G, L and B. Moreover, for all our computational hardness results, the instances of the problems arising from our reductions satisfy this constraint, which is guaranteed syntactically by our reduction.

In summary, in the rest of this paper, whenever we prove *an upper bound* for some computational problem, namely an upper bound on the number of steps or queries to the function oracle required to solve the problem in the black-box model, or the containment of the problem in some complexity class in the white-box model, we assume that the afore-described properties are satisfied by the O_f or C_f provided in the input. On the other hand, whenever we prove a *lower bound* for some computational problem, namely a lower bound on the number of steps/queries required to solve it in the black-box model, or its hardness for some complexity class in the white-box model, the instances arising in our lower bounds are guaranteed to satisfy the above properties syntactically by our constructions. As such, our hardness results will not exploit the difficulty in checking whether O_f or C_f satisfy the above constraints in order to infuse computational complexity into our problems, but will faithfully target the computational problems pertaining to min-max optimization of smooth and Lipschitz objectives that we aim to understand in this paper.

Complexity Classes and Reductions. Next we define the main complexity classes that we use in this paper, namely NP, FNP and PPAD, as well as the notion of reduction used to show containment or hardness of a problem for one of these complexity classes.

Definition 2.2 (Search Problems, NP, FNP)**.** A binary relation $Q \subseteq \{0,1\}^* \times \{0,1\}^*$ is in the class FNP if (i) for every $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^*$ such that $(\boldsymbol{x}, \boldsymbol{y}) \in Q$, it holds that $|\boldsymbol{y}| \leq \text{poly}(|\boldsymbol{x}|)$; and (ii) there exists an algorithm that verifies whether $(\boldsymbol{x}, \boldsymbol{y}) \in Q$ in time $\text{poly}(|\boldsymbol{x}|, |\boldsymbol{y}|)$. The *search problem* associated with a binary relation Q takes some $\boldsymbol{x}$ as input and requests as output some $\boldsymbol{y}$ such that $(\boldsymbol{x}, \boldsymbol{y}) \in Q$ or outputting $\perp$ if no such $\boldsymbol{y}$ exists. The *decision problem* associated with Q takes some $\boldsymbol{x}$ as input and requests as output the bit 1, if there exists some $\boldsymbol{y}$ such that $(\boldsymbol{x}, \boldsymbol{y}) \in Q$, and the bit 0, otherwise. The class NP is defined as the set of decision problems associated with relations $Q \in$ FNP.

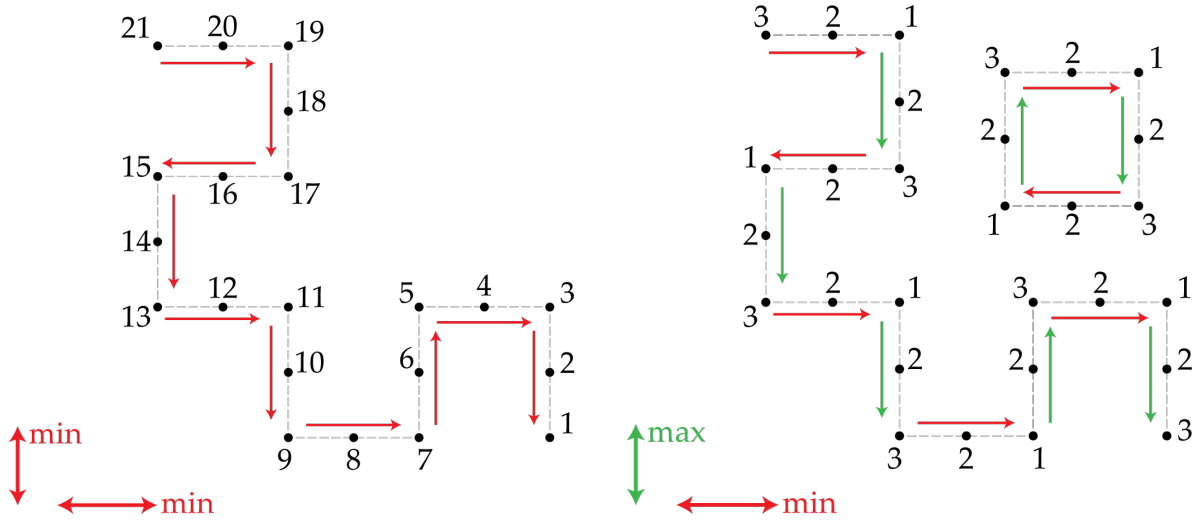
To define the complexity class PPAD we first define the notion of polynomial-time reductions between search problems[3], and the computational problem End-of-a-Line[4].

Definition 2.3 (Polynomial-Time Reductions)**.** A search problem P_1 is *polynomial-time reducible* to a search problem P_2 if there exist polynomial-time computable functions $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ with the following properties: (i) if $\boldsymbol{x}$ is an input to P_1, then $f(\boldsymbol{x})$ is an input to P_2; and (ii) if $\boldsymbol{y}$ is a solution to P_2 on input $f(\boldsymbol{x})$, then $g(\boldsymbol{x}, f(\boldsymbol{x}), \boldsymbol{y})$ is a solution to P_1 on input $\boldsymbol{x}$.

We next define the complete problem that describes the class PPAD.

End-of-a-Line.
Input: Binary circuits C_S (for successor) and C_P (for predecessor) with n inputs and n outputs.
Output: One of the following:
0. $\mathbf{0}$ if either both $C_P(C_S(\mathbf{0}))$ and $C_S(C_P(\mathbf{0}))$ are equal to $\mathbf{0}$, or if they are both different than $\mathbf{0}$, where $\mathbf{0}$ is the all-0 string.

[3] In this paper we only define and consider Karp-reductions between search problems.
[4] This problem is sometimes called End-of-the-Line, but we adopt the nomenclature proposed by [40] since we agree that it describes the problem better.



(a) Min-min problem; the function values reveal the location of the points within best response path.

(b) Min-max problem; the function values do not reveal the location of the points within best response path.

Figure 2: Long paths of best-response dynamics in min-min problems (Panel (a)) and min-max problems (Panel (b)), where horizontal moves correspond to one player (who is a minimizer in both (a) and (b)) and vertical moves correspond to the other player (who is minimizer in (a) but a maximizer in (b)). In Panels (a) and (b), we show the function value at a subset of discrete points in a 2D grid along a long path of best-response dynamics, where for our illustration we assumed that the derivative of the objective with respect to either variable always has absolute value 2. As we see in Panel (a), the function value at some point along a long path of the best-response dynamics in a min-min problem reveals information about where in the path that point lies. This is in sharp contrast to min-max problems where only local information is revealed about the objective as shown in Panel (b), due to the frequent turns of the path. In Panel (b) we also show that the best-response dynamics in min-max problems can form closed paths. This cannot happen in min-min problems as the function value must decrease along paths of best-response dynamics, and hence it is impossible in min-min problems to build long best-response paths with function values that can be computed locally.

1. a binary string $\boldsymbol{x} \in \{0,1\}^n$ such that $\boldsymbol{x} \neq \mathbf{0}$ and $C_P(C_S(\boldsymbol{x})) \neq \boldsymbol{x}$ or $C_S(C_P(\boldsymbol{x})) \neq \boldsymbol{x}$.

To make sense of the above definition, we envision that the circuits C_S and C_P implicitly define a directed graph, with vertex set $\{0,1\}^n$, such that the directed edge $(\boldsymbol{x}, \boldsymbol{y}) \in \{0,1\}^n \times \{0,1\}^n$ belongs to the graph if and only if $C_S(\boldsymbol{x}) = \boldsymbol{y}$ and $C_P(\boldsymbol{y}) = \boldsymbol{x}$. As such, all vertices in the implicitly defined graph have in-degree and out-degree at most 1. The above problem permits an output of $\mathbf{0}$ if $\mathbf{0}$ has equal in-degree and out-degree in this graph. Otherwise it permits an output $\boldsymbol{x} \neq \mathbf{0}$ such that $\boldsymbol{x}$ has in-degree or out-degree equal to 0. It follows by the parity argument on directed graphs, namely that in every directed graph the sum of in-degrees equals the sum of out-degrees, that End-of-a-Line is a *total problem*, i.e. that for any possible binary circuits C_S and C_P there exists a solution of the "0." kind or the "1." kind in the definition of our problem (or both). Indeed, if $\mathbf{0}$ has unequal in- and out-degrees, there must exist another vertex $\boldsymbol{x} \neq \mathbf{0}$ with unequal in- and out-degrees, thus one of these degrees must be 0 (as all vertices in the graph have in- and out-degrees bounded by 1). We are finally ready to define the complexity class PPAD introduced by [37].

Definition 2.4. The complexity class PPAD contains all search problems that are polynomial time reducible to the End-of-a-Line problem.

The complexity class PPAD is of particular importance, since it contains lots of fundamental problems in Game Theory, Economics, Topology and several other fields [11, 12]. A particularly important PPAD-complete problem is finding fixed points of continuous functions, whose existence is guaranteed by Brouwer's fixed point theorem.

Brouwer.
Input: Scalars L and γ and a polynomial-time Turing machine C_M evaluating a L-Lipschitz function $M : [0,1]^d \to [0,1]^d$.
Output: A point $\boldsymbol{z}^\star \in [0,1]^d$ such that $\left\|\boldsymbol{z}^\star - M(\boldsymbol{z}^\star)\right\|_2 < \gamma$.

While not stated exactly in this form, the following is a straightforward implication of the results presented in [9].

Lemma 2.5 ([9]). *Brouwer is* PPAD*-complete even when* $d = 2$. *Additionally, Brouwer is* PPAD*-complete even when* $\gamma = \text{poly}(1/d)$ *and* $L = \text{poly}(d)$.

Remark 2.6 (Respresentation of a polynomial-time Turing Machine). *In the definition of the problem Brouwer we assume that we are given in the input the description of a Turing Machine* C_M *that computes*

the map M. In order for polynomial-time reductions to and from this problem to be meaningful we need to have an upper bound on the running time of this Turing Machine which we want to be polynomial in the input of the Turing Machine. The formal way to ensure this and derive meaningful complexity results is to define a different problem, say k-Brouwer, for every $k \in \mathbb{N}$. In the problem k-Brouwer the input Turing Machine C_M has running time bounded by n^k in the size n of its input. In the rest of the paper whenever we say that a polynomial-time Turing Machine is required in the input to a computational problem Pr, we formally mean that we define a hierarchy of problems k-Pr, $k \in \mathbb{N}$, such that k-Pr takes as input Turing Machines with running time bounded by n^k, and we interpret computational complexity results for Pr in the following way: whenever we prove that Pr belongs to some complexity class, we prove that k-Pr belongs to the complexity class for all $k \in \mathbb{N}$; whenever we prove that Pr is hard for some complexity class, we prove that, for some absolute constant k_0 determined in the hardness proof, k-Pr is hard for that class, for all $k \geq k_0$. For simplicity of exposition of our problems and results we do not repeat this discussion in the rest of this paper.

3 COMPUTATIONAL PROBLEMS OF INTEREST

In this section, we define the computational problems that we study in this paper and discuss our main results, postponing formal statements to Section 4. We start in Section 3.1 by defining the mathematical objects of our study, and proceed in Section 3.2 to define our main computational problems, namely: (1) finding approximate stationary points; (2) finding approximate local minima; and (3) finding approximate local min-max equilibria. In Section 3.3, we present some bonus problems, which are intimately related, as we will see, to problems (2) and (3). As discussed in Section 2, for ease of presentation, we define our problems as promise problems.

3.1 Mathematical Definitions

We define the concepts of *stationary points*, *local minima*, and *local min-max equilibria* of real valued functions, and make some remarks about their existence, as well as their computational complexity. The formal discussion of the latter is postponed to Sections 3.2 and 4.

Before we proceed with our definitions, recall that the goal of this paper is to study constrained optimization. Our domain will be the hypercube $[0,1]^d$, which we might intersect with the set $\{\boldsymbol{x} \mid \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}\}$, for some convex (potentially multivariate) function $\boldsymbol{g}$. Although most of the definitions and results that we explore in this paper can be extended to arbitrary convex functions, we will focus on the case where $\boldsymbol{g}$ is linear, and the feasible set is thus a polytope. Focusing on this case avoids additional complications related to the representation of $\boldsymbol{g}$ in the input to the computational problems that we define in the next section, and avoids also issues related to verifying the convexity of $\boldsymbol{g}$.

Definition 3.1 (Feasible Set and Refutation of Feasibility)**.** Given $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$, we define the set of feasible solutions to be $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) = \{\boldsymbol{z} \in [0,1]^d \mid \boldsymbol{A}^T \boldsymbol{z} \leq \boldsymbol{b}\}$. Observe that testing whether $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is empty can be done in polynomial time in the bit complexity of $\boldsymbol{A}$ and $\boldsymbol{b}$.

Definition 3.2 (Projection Operator)**.** For a nonempty, closed, and convex set $K \subset \mathbb{R}^d$, we define the projection operator $\Pi_K : \mathbb{R}^d \to K$ as follows $\Pi_K \boldsymbol{x} = \operatorname{argmin}_{\boldsymbol{y} \in K} \|\boldsymbol{x} - \boldsymbol{y}\|_2$. It is well-known that for any nonempty, closed, and convex set K the $\operatorname{argmin}_{\boldsymbol{y} \in K} \|\boldsymbol{x} - \boldsymbol{y}\|_2$ exists and is unique, hence Π_K is well defined.

Now that we have defined the domain of the real-valued functions that we consider in this paper we are ready to define a notion of approximate stationary points.

Definition 3.3 (ε-Stationary Point)**.** Let $f : [0,1]^d \to \mathbb{R}$ be a G-Lipschitz and L-smooth function and $\boldsymbol{A} \in \mathbb{R}^{d \times m}$, $\boldsymbol{b} \in \mathbb{R}^m$. We call a point $\boldsymbol{x}^\star \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ a *ε-stationary point* of f if $\left\|\nabla f(\boldsymbol{x}^\star)\right\|_2 < \varepsilon$.

It is easy to see that there exist continuously differentiable functions f that do not have any (approximate) stationary points, e.g. linear functions. As we will see later in this paper, deciding whether a given function f has a stationary point is NP-hard and, in fact, it is even NP-hard to decide whether a function has an approximate stationary point of a very gross approximation. At the same time, verifying whether a given point is (approximately) stationary can be done efficiently given access to a polynomial-time Turing machine that computes ∇f, so the problem of deciding whether an (approximate) stationary point exists lies in NP, as long as we can guarantee that, if there is such a point, there will also be one with polynomial bit complexity. We postpone a formal discussion of the computational complexity of finding (approximate) stationary points or deciding their existence until we have formally defined our corresponding computational problem and settled the bit complexity of its solutions.

For the definition of local minima and local min-max equilibria we need the notion of closed d-dimensional Euclidean balls.

Definition 3.4 (Euclidean Ball)**.** For $r \in \mathbb{R}_+$, the *closed Euclidean ball of radius r* is the set $\mathrm{B}_d(r) = \left\{\boldsymbol{x} \in \mathbb{R}^d \mid \|\boldsymbol{x}\|_2 \leq r\right\}$. We also define the *closed Euclidean ball of radius r centered at $\boldsymbol{z} \in \mathbb{R}^d$* to be the set $\mathrm{B}_d(r; \boldsymbol{z}) = \left\{\boldsymbol{x} \in \mathbb{R}^d \mid \|\boldsymbol{x} - \boldsymbol{z}\|_2 \leq r\right\}$.

Definition 3.5 ((ε, δ)-Local Minimum)**.** Let $f : [0,1]^d \to \mathbb{R}$ be a G-Lipschitz and L-smooth function, $\boldsymbol{A} \in \mathbb{R}^{d \times m}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\varepsilon, \delta > 0$. A point $\boldsymbol{x}^\star \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is an *(ε, δ)-local minimum* of f constrained on $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ if and only if $f(\boldsymbol{x}^\star) < f(\boldsymbol{x}) + \varepsilon$ for every $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\boldsymbol{x} \in \mathrm{B}_d(\delta; \boldsymbol{x}^\star)$.

To be clear, using the term "local minimum" in Definition 3.5 is a bit of a misnomer, since for large enough values of δ the definition captures global minima as well. As δ ranges from large to small, our notion of (ε, δ)-local minimum transitions from being an ε-globally optimal point to being an ε-locally optimal point. Importantly, unlike (approximate) stationary points, a (ε, δ)-local minimum is guaranteed to exist for all $\varepsilon, \delta > 0$ due to the compactness of $[0,1]^d \cap \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ and the continuity of f. Thus the problem of finding an (ε, δ)-local minimum is *total* for arbitrary values of ε and δ. On the negative side, for arbitrary values of ε and δ, there is no polynomial-size and polynomial-time verifiable witness for certifying that a point $\boldsymbol{x}^\star$ is an (ε, δ)-local minimum. Thus the problem of finding an (ε, δ)-local minimum is not known to lie in FNP. As we will see in Section 4, this issue can be circumvented

if we focus on particular settings of ε and δ, in relationship to the Lipschitzness and smoothness of f and the dimension d.

Finally we define (ε, δ)-local min-max equilibrium as follows, recasting Definition 1.1 to the constraint set $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$.

Definition 3.6 ((ε, δ)-Local Min-Max Equilibrium)**.** Let f such that $f : [0,1]^{d_1} \times [0,1]^{d_2} \to \mathbb{R}$ be a G-Lipschitz and L-smooth function, $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$, where $d = d_1 + d_2$, and $\varepsilon, \delta > 0$. A point $(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is an (ε, δ)-*local min-max equilibrium* of f if and only if the following hold:

- $f(\boldsymbol{x}^\star, \boldsymbol{y}^\star) < f(\boldsymbol{x}, \boldsymbol{y}^\star) + \varepsilon$ for every $\boldsymbol{x} \in \mathrm{B}_{d_1}(\delta; \boldsymbol{x}^\star)$ with $(\boldsymbol{x}, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$; and
- $f(\boldsymbol{x}^\star, \boldsymbol{y}^\star) > f(\boldsymbol{x}^\star, \boldsymbol{y}) - \varepsilon$ for every $\boldsymbol{y} \in \mathrm{B}_{d_2}(\delta; \boldsymbol{y}^\star)$ with $(\boldsymbol{x}^\star, \boldsymbol{y}) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$.

Similarly to Definition 3.5, for large enough values of δ, Definition 3.6 captures global min-max equilibria as well. As δ ranges from large to small, our notion of (ε, δ)-local min-max equilibrium transitions from being an ε-approximate min-max equilibrium to being an ε-approximate local min-max equilibrium. Moreover, in comparison to local minima and stationary points, the problem of finding an (ε, δ)-local min-max equilibrium is neither total nor can its solutions be verified efficiently for all values of ε and δ, even when $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) = [0,1]^d$. Again, this issue can be circumvented if we focus on particular settings of ε and δ values, as we will see in Section 4.

3.2 First-Order Local Optimization Computational Problems

In this section, we define the search problems associated with our aforementioned definitions of approximate stationary points, local minima, and local min-max equilibria. We state our problems in terms of white-box access to the function f and its gradient. Switching to the black-box variants of our computational problems amounts to simply replacing the Turing machines provided in the input of the problems with oracle access to the function and its gradient, as discussed in Section 2. As per our discussion in the same section, we define our computational problems as *promise problems*, the promise being that the Turing machine (or oracle) provided in the input to our problems outputs function values and gradient values that are consistent with a smooth and Lipschitz function with the prescribed in the input smoothness and Lipschitzness. Besides making the presentation cleaner, as we discussed in Section 2, the motivation for doing so is to prevent the possibility that computational complexity is tacked into our problems due to the possibility that the Turing machines/oracles provided in the input do not output function and gradient values that are consistent with a Lipschitz and smooth function. Importantly, all our computational hardness results syntactically guarantee that the Turing machines/oracles provided as input to our constructed hard instances satisfy these constraints.

Before stating our main computational problems below, we note that, for each problem, the dimension d (in unary representation) is also an implicit input, as the description of the Turing machine C_f (or the interface to the oracle O_f in the black-box counterpart of each problem below) has size at least linear in d. We also refer to Remark 2.6 for how we may formally study complexity problems that take a polynomial-time Turing Machine in their input.

StationaryPoint.
Input: Scalars $\varepsilon, G, L, B > 0$ and a polynomial-time Turing machine C_f evaluating a G-Lipschitz and L-smooth function $f : [0,1]^d \to [-B, B]$ and its gradient $\nabla f : [0,1]^d \to \mathbb{R}^d$; a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ such that $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \neq \emptyset$.
Output: If there exists some point $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\|\nabla f(\boldsymbol{x})\|_2 < \varepsilon/2$, output some point $\boldsymbol{x}^\star \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\left\|\nabla f(\boldsymbol{x}^\star)\right\|_2 < \varepsilon$; if, for all $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$, $\|\nabla f(\boldsymbol{x})\|_2 > \varepsilon$, output $\perp$; otherwise, it is allowed to either output $\boldsymbol{x}^\star \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\left\|\nabla f(\boldsymbol{x}^\star)\right\|_2 < \varepsilon$ or to output $\perp$.

It is easy to see that StationaryPoint lies in FNP. Indeed, if there exists some point $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\|\nabla f(\boldsymbol{x})\|_2 < \varepsilon/2$, then by the L-smoothness of f there must exist some point $\boldsymbol{x}^\star \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ of bit complexity polynomial in the size of the input such that $\left\|\nabla f(\boldsymbol{x}^\star)\right\|_2 < \varepsilon$. On the other hand, it is clear that no such point exists if for all $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$, $\|\nabla f(\boldsymbol{x})\|_2 > \varepsilon$. We note that the looseness of the output requirement in our problem for functions f that do not have points $x \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\|\nabla f(\boldsymbol{x})\|_2 < \varepsilon/2$ but do have points $x \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\|\nabla f(\boldsymbol{x})\|_2 \leq \varepsilon$ is introduced for the sole purpose of making the problem lie in FNP, as otherwise we would not be able to guarantee that the solutions to our search problem have polynomial bit complexity. As we show in Section 4, StationaryPoint is also FNP-hard, even when ε is a constant, the constraint set is very simple, namely $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) = [0,1]^d$, and G, L are both polynomial in d.

Next, we define the computational problems associated with local minimum and local min-max equilibrium. Recall that the first is guaranteed to have a solution, because, in particular, a global minimum exists due to the continuity of f and the compactness of $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$.

LocalMin.
Input: Scalars $\varepsilon, \delta, G, L, B > 0$ and a polynomial-time Turing machine C_f evaluating a G-Lipschitz and L-smooth function $f : [0,1]^d \to [-B, B]$ and its gradient $\nabla f : [0,1]^d \to \mathbb{R}^d$; a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ such that $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \neq \emptyset$.
Output: A point $\boldsymbol{x}^\star \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $f(\boldsymbol{x}^\star) < f(\boldsymbol{x}) + \varepsilon$ for all $\boldsymbol{x} \in \mathrm{B}_d(\delta; \boldsymbol{x}^\star) \cap \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$.

LocalMinMax.
Input: Scalars $\varepsilon, \delta, G, L, B > 0$; a polynomial-time Turing machine C_f evaluating a G-Lipschitz and L-smooth function $f : [0,1]^{d_1} \times [0,1]^{d_2} \to [-B, B]$ and its gradient $\nabla f : [0,1]^{d_1} \times [0,1]^{d_2} \to \mathbb{R}^{d_1+d_2}$; a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ such that $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \neq \emptyset$, where $d = d_1 + d_2$.
Output: A point $(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that
- $f(\boldsymbol{x}^\star, \boldsymbol{y}^\star) < f(\boldsymbol{x}, \boldsymbol{y}^\star) + \varepsilon$ for all $\boldsymbol{x} \in B_{d_1}(\delta; \boldsymbol{x}^\star)$ with $(\boldsymbol{x}, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ and
- $f(\boldsymbol{x}^\star, \boldsymbol{y}^\star) > f(\boldsymbol{x}^\star, \boldsymbol{y}) - \varepsilon$ for all $\boldsymbol{y} \in B_{d_2}(\delta; \boldsymbol{y}^\star)$ with $(\boldsymbol{x}^\star, \boldsymbol{y}) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$,

or $\perp$ if no such point exists.

Unlike the problem StationaryPoint the problems LocalMin and LocalMinMax exhibit vastly different behavior, depending on the

values of the inputs ε and δ in relationship to G, L and d, as we will see in Section 4 where we summarize our computational complexity results. This range of behaviors is rooted at our earlier remark that, depending on the value of δ provided in the input to these problems, they capture the complexity of finding *global* minima/min-max equilibria, for large values of δ, as well as finding *local* minima/min-max equilibria, for small values of δ.

3.3 Bonus Problems: Fixed Points of Gradient Descent/Gradient Descent-Ascent

Next we present a couple of bonus problems, GDFixedPoint and GDAFixedPoint, which respectively capture the computation of fixed points of the (projected) gradient descent and the (projected) gradient descent-ascent dynamics, with learning rate = 1. These problems are intimately related, indeed equivalent under polynomial-time reductions, to problems LocalMin and LocalMinMax respectively, in certain regimes of the approximation parameters. Before stating problems GDFixedPoint and GDAFixedPoint, we define the mappings F_{GD} and F_{GDA} whose fixed points these problems are targeting.

Definition 3.7 (Projected Gradient Descent)**.** For a closed and convex $K \subseteq \mathbb{R}^d$ and some continuously differentiable function $f : K \to \mathbb{R}$, we define the *Projected Gradient Descent Dynamics with learning rate* 1 as the map $F_{GD} : K \to K$, where $F_{GD}(\boldsymbol{x}) = \Pi_K(\boldsymbol{x} - \nabla f(\boldsymbol{x}))$.

Definition 3.8 (Projected Gradient Descent/Ascent)**.** For a closed and convex $K \subseteq \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$ and some continuously differentiable function $f : K \to \mathbb{R}$, we define the *Unsafe Projected Gradient Descent/Ascent Dynamic with learning rate* 1 as the map $F_{GDA} : K \to \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$ defined as follows

$$F_{GDA}(\boldsymbol{x}, \boldsymbol{y}) \triangleq \begin{bmatrix} \Pi_{K(\boldsymbol{y})}(\boldsymbol{x} - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})) \\ \Pi_{K(\boldsymbol{x})}(\boldsymbol{y} + \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})) \end{bmatrix} \triangleq \begin{bmatrix} F_{GDAx}(\boldsymbol{x}, \boldsymbol{y}) \\ F_{GDAy}(\boldsymbol{x}, \boldsymbol{y}) \end{bmatrix}$$

for all $(\boldsymbol{x}, \boldsymbol{y}) \in K$, where $K(\boldsymbol{y}) = \{\boldsymbol{x}' \mid (\boldsymbol{x}', \boldsymbol{y}) \in K\}$ and $K(\boldsymbol{x}) = \{\boldsymbol{y}' \mid (\boldsymbol{x}, \boldsymbol{y}') \in K\}$.

Note that F_{GDA} is called "unsafe" because the projection happens individually for $\boldsymbol{x} - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{y} + \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$, thus $F_{GDA}(\boldsymbol{x}, \boldsymbol{y})$ may not lie in K. We also define the "safe" version F_{sGDA}, which projects the pair $(\boldsymbol{x} - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{y} + \nabla_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y}))$ jointly onto K. In the full version of the paper we show that computing fixed points of F_{GDA} and F_{sGDA} are computationally equivalent so we stick to F_{GDA} which makes the presentation slightly cleaner.

We are now ready to define GDFixedPoint and GDAFixedPoint. As per earlier discussions, we define these computational problems as *promise problems*, the promise being that the Turing machine provided in the input to these problems outputs function values and gradient values that are consistent with a smooth and Lipschitz function with the prescribed, in the input to these problems, smoothness and Lipschitzness.

GDFixedPoint.
Input: Scalars $\alpha, G, L, B > 0$ and a polynomial-time Turing machine C_f evaluating a G-Lipschitz and L-smooth function $f : [0,1]^d \to [-B, B]$ and its gradient $\nabla f : [0,1]^d \to \mathbb{R}^d$; a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ such that $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \neq \emptyset$.
Output: A point $\boldsymbol{x}^\star \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\left\|\boldsymbol{x}^\star - F_{GD}(\boldsymbol{x}^\star)\right\|_2 < \alpha$, where $K = \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is the projection set used in the definition of F_{GD}.

GDAFixedPoint.
Input: Scalars $\alpha, G, L, B > 0$ and a polynomial-time Turing machine C_f evaluating a G-Lipschitz and L-smooth function $f : [0,1]^{d_1} \times [0,1]^{d_2} \to [-B, B]$ and its gradient $\nabla f : [0,1]^{d_1} \times [0,1]^{d_2} \to \mathbb{R}^{d_1+d_2}$; a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ such that $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \neq \emptyset$, where $d = d_1 + d_2$.
Output: A point $(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ such that $\left\|(\boldsymbol{x}^\star, \boldsymbol{y}^\star) - F_{GDA}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)\right\|_2 < \alpha$, where $K = \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is the projection set used in the definition of F_{GDA}.

In the full version of the paper we show that GDFixedPoint and LocalMin are equivalent under polynomial-time reductions, and the problems GDAFixedPoint and LocalMinMax are equivalent under polynomial-time reductions, in certain regimes of the approximation parameters.

4 SUMMARY OF RESULTS

In this section we summarize our results for the optimization problems that we defined in the previous section. We start with our theorem about the complexity of finding approximate stationary points, which we show to be FNP-complete even for large values of the approximation.

Theorem 4.1 (Complexity of Approximate Stationary Points)**.** *The computational problem* StationaryPoint *is* FNP*-complete, even when ε is set to any value $\leq 1/24$, and even when $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) = [0,1]^d$, $G = \sqrt{d}$, $L = d$, and $B = 1$.*

It is folklore and easy to verify that approximate stationary points always exist and can be found in time $\text{poly}(B, 1/\varepsilon, L)$ when the domain of f is unconstrained, i.e. it is the whole $\mathbb{R}^d$, and the range of f is bounded, i.e., when $f(\mathbb{R}^d) \subseteq [-B, B]$. Theorem 4.1 implies that such a guarantee should not be expected in the bounded domain case, where the existence of approximate stationary points is not guaranteed and must also be verified. In particular, it follows from our theorem that any algorithm that verifies the existence of and computes approximate stationary points in the constrained case should take time that is super-polynomial in at least one of G, L, or d, unless P = NP. The proof of Theorem 4.1 is based on an elegant construction for converting (real valued) stationary points of an appropriately constructed function to (binary) solutions of a target Sat instance. This conversion involves the use of Lovász Local Lemma [17].

The complexity of LocalMin and LocalMinMax is more difficult to characterize, as the nature of these problems changes drastically depending on the relationship of δ with with ε, G, L and d, which determines whether these problems ask for a *globally* vs *locally* approximately optimal solution. In particular, there are two regimes wherein the complexity of both problems is simple to characterize.

- **Global Regime.** When $\delta \geq \sqrt{d}$ then both LocalMin and LocalMinMax ask for a *globally* optimal solution. In this regime it is not difficult to see that both problems are FNP-hard to solve even when $\varepsilon = \Theta(1)$ and G, L are $O(d)$.

- **Trivial Regime.** When δ satisfies $\delta < \varepsilon/G$, then for every point $\boldsymbol{z} \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ it holds that $|f(\boldsymbol{z}) - f(\boldsymbol{z}')| < \varepsilon$ for every $\boldsymbol{z}' \in \mathrm{B}_d(\delta; \boldsymbol{z})$ with $\boldsymbol{z}' \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$. Thus, every point $\boldsymbol{z}$ in the domain $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is a solution to both LocalMin and LocalMinMax.

It is clear from our discussion above, and in earlier sections, that, to really capture the complexity of finding local as opposed to global minima/min-max equilibria, we should restrict the value of δ. We identify the following regime, which we call the "*local regime*." As we argue shortly, this regime is markedly different from the global regime identified above in that (i) a solution is guaranteed to exist for both our problems of interest, where in the global regime only LocalMin is guaranteed to have a solution; and (ii) their computational complexity transitions to lower complexity classes.

- **Local Regime.** Our main focus in this paper is the regime defined by $\delta < \sqrt{2\varepsilon/L}$. In this regime it is well known that Projected Gradient Descent can solve LocalMin in time $O(B \cdot L/\varepsilon)$. Our main interest is understanding the complexity of LocalMinMax, which is not well understood in this regime. We note that the use of the constant 2 in the constraint $\delta < \sqrt{2\varepsilon/L}$ which defines the local regime has a natural motivation: consider a point $\boldsymbol{z}$ where a L-smooth function f has $\nabla f(\boldsymbol{z}) = 0$; it follows from the definition of smoothness that $\boldsymbol{z}$ is both an (ε, δ)-local min and an (ε, δ)-local min-max equilibrium, as long as $\delta < \sqrt{2\varepsilon/L}$.

The following theorems provide tight upper and lower bounds on the computational complexity of solving LocalMinMax in the local regime. For compactness, we define the following problem:

Definition 4.2 (Local Regime LocalMinMax)**.** We define the *local-regime local min-max equilibrium computation problem*, in short LR-LocalMinMax, to be the search problem LocalMinMax restricted to instances in the local regime, i.e. satisfying $\delta < \sqrt{2\varepsilon/L}$.

Theorem 4.3 (Existence of Approximate Local Min-Max Equilibrium)**.** *The computational problem* LR-LocalMinMax *belongs to* PPAD. *As a byproduct, if some function f is G-Lipschitz and L-smooth, then an (ε, δ)-local min-max equilibrium is guaranteed to exist when $\delta < \sqrt{2\varepsilon/L}$, i.e. in the local regime.*

Theorem 4.4 (Hardness of Finding Approximate Local Min-Max Equilibrium)**.** *The search problem* LR-LocalMinMax *is* PPAD*-hard, for any $\delta \geq \sqrt{\varepsilon/L}$, and even when it holds that $1/\varepsilon = \mathrm{poly}(d)$, $G = \mathrm{poly}(d)$, $L = \mathrm{poly}(d)$, and $B = d$.*

Theorem 4.4 implies that any algorithm that computes an (ε, δ)-local min-max equilibrium of a G-Lipschitz and L-smooth function f in the local regime should take time that is super-polynomial in at least one of $1/\varepsilon$, G, L or d, unless FP = PPAD. As such, the complexity of computing local min-max equilibria in the local regime is markedly different from the complexity of computing local minima, which can be found using Projected Gradient Descent in $\mathrm{poly}(G, L, 1/\varepsilon, d)$ time and function/gradient evaluations.

An important property of our reduction in the proof of Theorem 4.4 is that it is a *black-box reduction*. We can hence prove the following unconditional lower bound in the black-box model.

Theorem 4.5 (Black-Box Lower Bound for Finding Approximate Local Min-Max Equilibrium)**.** *Suppose $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$ are given together with an oracle O_f that outputs a G-Lipschtz and L-smooth function $f : \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \to [-1, 1]$ and its gradient ∇f. Let also $\delta \geq \sqrt{L/\varepsilon}$, $\varepsilon \leq G^2/L$, and let all the parameters $1/\varepsilon$, $1/\delta$, L, G be upper bounded by* $\mathrm{poly}(d)$*. Then any algorithm that has access to f only through O_f and computes an (ε, δ)-local min-max equilibrium has to make a number of queries to O_f that is exponential in at least one of the parameters: $1/\varepsilon$, G, L or d even when $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \subseteq [0, 1]^d$.*

ACKNOWLEDGEMENTS

This work was supported by NSF Awards IIS-1741137, CCF-1617730 and CCF-1901292, by a Simons Investigator Award, by the DOE PhILMs project (No. DE-AC05-76RL01830), and by the DARPA award HR00111990021. M.Z. was also supported by Google Ph.D. Fellowship. S.S. was supported by NRF 2018 Fellowship NRF-NRFF2018-07.

REFERENCES

[1] Jacob Abernethy, Kevin A Lai, and Andre Wibisono. 2019. Last-iterate convergence rates for min-max optimization.
[2] Ilan Adler. 2013. The equivalence of linear programs and zero-sum games. *International Journal of Game Theory* 42, 1 (2013), 165–177. https://doi.org/10.1007/s00182-012-0328-8
[3] Leonard Adolphs, Hadi Daneshmand, Aurelien Lucchi, and Thomas Hofmann. 2019. Local saddle point optimization: A curvature exploitation approach. *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), 486–495.
[4] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. 2017. Finding approximate local minima faster than gradient descent. *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (2017), 1195–1199.
[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), 214–223.
[6] David Blackwell. 1956. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.* 6, 1 (1956), 1–8. https://doi.org/10.2140/pjm.1956.6.1
[7] Sébastien Bubeck and Nicolo Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5, 1 (2012), 1–122. https://doi.org/10.1561/2200000024
[8] Nikolo Cesa-Bianchi and Gabor Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press.
[9] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. 2009. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)* 56, 3 (2009), 1–57.
[10] George B. Dantzig. 1951. A proof of the equivalence of the programming problem and the game probslem. *Koopmans, T. C., editor(s), Activity Analysis of Production and Allocation* (1951).
[11] Constantinos Daskalakis. 2018. Equilibria, Fixed Points, and Computational Complexity - Nevanlinna Prize Lecture. *Proceedings of the International Congress of Mathematicians (ICM)* 1 (2018), 147–209.
[12] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. 2009. The complexity of computing a Nash equilibrium. *SIAM J. Comput.* 39, 1 (2009), 195–259.
[13] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. 2018. Training GANs with Optimism.. In *International Conference on Learning Representations (ICLR 2018)*.
[14] Constantinos Daskalakis and Ioannis Panageas. 2018. The limit points of (optimistic) gradient descent in min-max optimization. In *Advances in Neural Information Processing Systems*. 9236–9246.
[15] Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis. 2020. The complexity of constrained min-max optimization.
[16] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, Jul (2011), 2121–2159.
[17] Paul Erdős and László Lovász. 1973. Problems and results on 3-chromatic hypergraphs and some related questions. In *Colloquia Mathematica Societatis Janos Bolyai 10. Infinite and Finite Sets, Keszthely (Hungary)*. Citeseer.
[18] Francisco Facchinei and Jong-Shi Pang. 2007. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media.

[19] Ian Goodfellow. 2016. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160* (2016).

[20] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. 2672–2680.

[21] Elad Hazan. 2016. Introduction to online convex optimization. *Foundations and Trends® in Optimization* 2, 3-4 (2016), 157–325.

[22] M. D. Hirsch, C. H. Papadimitriou, and S. A. Vavasis. 1989. Exponential lower bounds for finding Brouwer fixed points. *Journal of Complexity* 5 (1989), 379–416.

[23] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. 2017. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1724–1732.

[24] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. 2019. What is Local Optimality in Nonconvex-Nonconcave Minimax Optimization? *arXiv preprint arXiv:1902.00618* (2019).

[25] David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. 1988. How easy is local search? *Journal of computer and system sciences* 37, 1 (1988), 79–100.

[26] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[27] GM Korpelevich. 1976. The extragradient method for finding saddle points and other problems. *Matecon* 12 (1976), 747–756.

[28] Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. 2019. First-order methods almost always avoid strict saddle points. *Math. Program.* 176, 1-2 (2019), 311–337. https://doi.org/10.1007/s10107-019-01374-3

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

[30] Eric Mazumdar and Lillian J Ratliff. 2018. On the convergence of gradient-based learning in continuous games. *arXiv preprint arXiv:1804.05464* (2018).

[31] N Meggido and CH Papadimitriou. 1989. *A note on total functions, existence theorems, and computational complexity.* Technical Report. Tech. report, IBM.

[32] Panayotis Mertikopoulos, Christos H. Papadimitriou, and Georgios Piliouras. 2018. Cycles in Adversarial Regularized Learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[33] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. 2018. Which Training Methods for GANs do actually Converge?. In *International Conference on Machine Learning*. 3481–3490.

[34] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. 2016. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163* (2016).

[35] Arkadi Nemirovski. 2004. Interior point polynomial time methods in convex programming. *Lecture notes* (2004).

[36] Arkadiĭ Semenovich Nemirovsky and David Borisovich Yudin. 1983. *Problem complexity and method efficiency in optimization.* Chichester: Wiley.

[37] Christos H Papadimitriou. 1994. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and system Sciences* 48, 3 (1994), 498–532.

[38] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the Convergence of Adam and Beyond. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

[39] J Ben Rosen. 1965. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society* (1965), 520–534.

[40] Aviad Rubinstein. 2016. Settling the complexity of computing approximate two-player Nash equilibria. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 258–265.

[41] Shai Shalev-Shwartz. 2012. Online learning and online convex optimization. *Foundations and Trends in Machine Learning* 4, 2 (2012), 107–194. https://doi.org/10.1561/2200000018

[42] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms.* Cambridge university press.

[43] John von Neumann. 1928. Zur Theorie der Gesellschaftsspiele. In *Math. Ann.* 295–320.