The Complexity of Constrained Min-Max Optimization

Constantinos Daskalakis
MIT
costis@csail.mit.edu

Stratis Skoulakis
SUTD
efstratios@sutd.edu.sg

Manolis Zampetakis
MIT
mzampet@mit.edu

September 22, 2020

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 even to approximate local min-max points, but the computational complexity of identifying them is also poorly understood. In this paper, we provide a characterization of the computational complexity of the problem, as well as of the limitations of first-order methods in constrained min-max optimization problems with nonconvex-nonconcave objectives and linear constraints.

As a warm-up, we show that, even when the objective is a Lipschitz and smooth differentiable function, deciding whether a min-max point exists, in fact even deciding whether an approximate min-max point exists, is NP-hard. More importantly, we show that an approximate local min-max point of large enough approximation is guaranteed to exist, but finding one such point is PPAD-complete. The same is true of computing an approximate fixed point of the (Projected) Gradient Descent/Ascent update dynamics.

An important byproduct of our proof is to establish an unconditional hardness result in the Nemirovsky-Yudin [NY83] oracle optimization model. We show that, given oracle access to some function $f : \mathcal{P} \to [-1, 1]$ and its gradient ∇f, where $\mathcal{P} \subseteq [0, 1]^d$ is a known convex polytope, every algorithm that finds a ε-approximate local min-max point needs to make a number of 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 and d is the dimension. 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.

Contents

1 Introduction

Min-Max Optimization has played a central role in the development of Game Theory [vN28], Convex Optimization [Dan51, Adl13], and Online Learning [Bla56, CBL06, SS12, BCB12, SSBD14, Haz16]. In its general constrained form, it can be written down as follows:

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

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 $g(\boldsymbol{x},\boldsymbol{y}) \leq 0$ 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., $g(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \leq 0$, that satisfies the following

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

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

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 [vN28, Ros65], 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. [Kor76, Nem04].[2] When the function is strongly convex-strongly concave, the rate becomes geometric [FP07].

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. [GPM+14, ACB17]), and robustifying deep neural network-based models against adversarial attacks (see e.g. [MMS+18]). These applications are 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. [Goo16, MPPSD16, DISZ18, MGN18, DP18, MR18, MPP18, ADLH19]. This experience is in stark contrast to minimization (resp. maximization) problems, where even for

[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 $g(\boldsymbol{x},\boldsymbol{y}) \leq 0$. 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 $\varepsilon(L,T)$.

nonconvex (resp. nonconcave) objectives, first-order methods have been found to efficiently converge to approximate local optima or stationary points (see e.g. [AAZB+17, JGN+17, LPP+19]), while practical methods such Stochastic Gradient Descent, Adagrad, and Adam [DHS11, KB14, RKK18] 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 hand side of (1.2) and (1.3) by some $\varepsilon > 0$.

▷ *Nonconvex-Nonconcave Objective.* By contrapositive, 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 there is an approximate min-max solution is NP-hard, even when the function is Lispchitz and smooth and the desired approximation error is an absolute constant (see Theorem 10.1).

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 [ALW19]; 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 [DP18, MR18, JNJ19]. We consider a generalization of the concept of local min-max equilibria, proposed in [DP18, MR18], 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. $g(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \leq 0$, and satisfies:

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

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

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 (Theorem 5.1) 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 (Theorem 5.2).

► **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 if ε is an absolute constant (Theorem 10.1).

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:

Informal Theorem 1 (see Theorems 4.3, 4.4 and 5.1)**.** *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 the constraint set is a polytope that is a subset of $[0,1]^d$, the objective takes values in $[-d,d]$ 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 [NY83] oracle optimization model, wherein we are given black-box access to oracles computing the objective function and its gradient. Our second main result is informally stated in Informal Theorem 2.

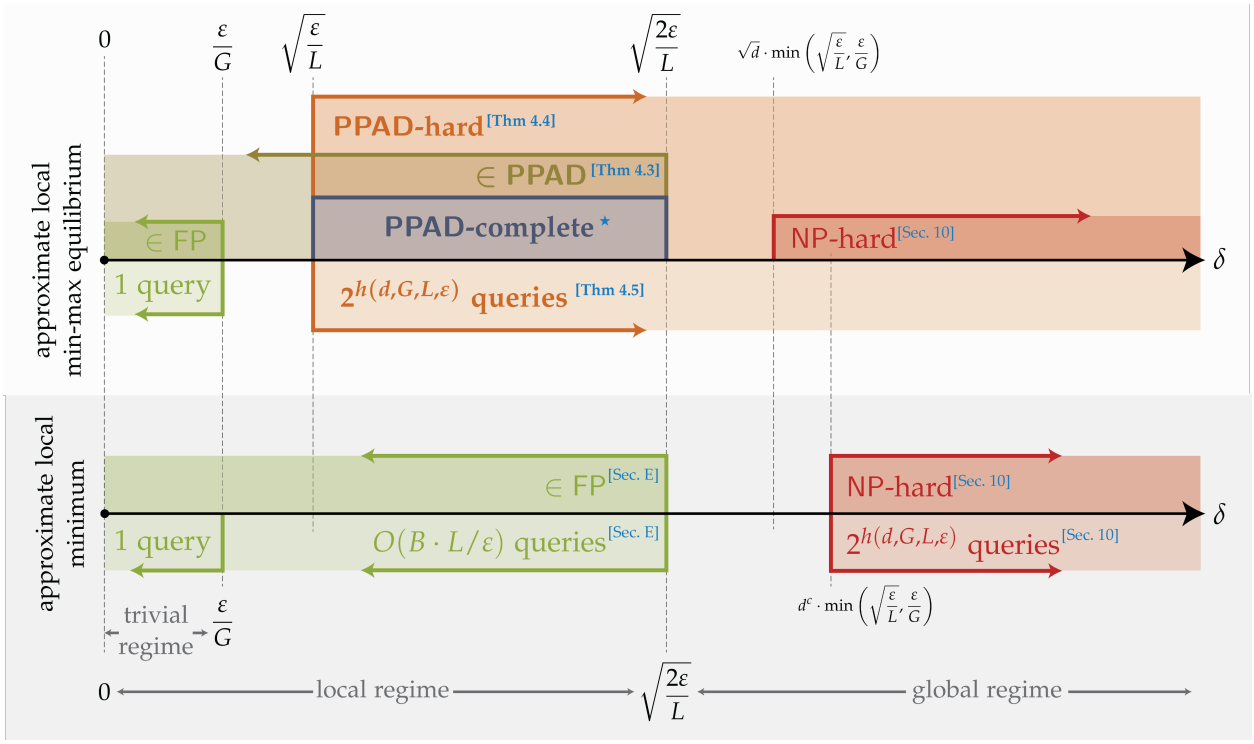


Figure 1: Overview of the results proven in this paper and comparison between the complexity of computing an (ε, δ)-approximate local minimum and an (ε, δ)-approximate local min-max equilibrium of a G-Lipschitz and L-smooth function over a d-dimensional polytope taking values in the interval $[-B, B]$. 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 $p \in \mathbb{R}_+$. With $\star$ we indicate our PPAD-completeness result which directly follows from Theorems 4.3 and 4.4. The NP-hardess results in the global regime are presented in Section 10. Finally, the folklore result showing the tractability of finding approximate local minima is presented for completeness of exposition in Appendix E. The claimed results for the trivial regime follow from the definition of Lipschitzness.

Informal 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 (see Section E). 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 Brief Overview of the Techniques

We very briefly outline some of the main ideas for the PPAD-hardness proof that we present in Sections 6 and 7. 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 useful for showing our hardness results. BiSperner is closely related to the problem of finding panchromatic simplices guaranteed by Sperner's lemma except, roughly speaking, that the vertices of the simplicization of a d-dimensional hypercube are colored with $2d$ rather than $d + 1$ colors, every point of the simplicization is colored with d colors rather than one, and we are seeking a vertex of the simplicization so that the union of colors on the vertices in its neighborhood covers the full set of colors. The first step of our proof is to show that BiSperner is PPAD-hard. This step follows from the hardness of computing Brouwer fixed points.

The step that we describe next is only implicitly done by our proof, but it serves as useful intuition for reading and understanding it. We want to define a *discrete* two-player zero-sum game whose local equilibrium points correspond to solutions of a given BiSperner instance. Our two players, called "minimizer" and "maximizer," each choose a vertex of the simplicization of the BiSperner instance. For every pair of strategies in our discrete game, i.e. vertices, chosen by our players, we define a function value and gradient values. Note that, at this point, we treat these values at different vertices of the simplicization as independent choices, i.e. are not defining a function over the continuum whose function values and gradient values are consistent with these choices. It is our intention, however, that in the *continuous* two-player zero-sum game that we obtain in the next paragraph via our interpolation scheme, wherein the minimizer and maximizer may choose any point in the continuous hypercube, the function value determines the payment of the minimizer to the maximizer, and the gradient value determines the direction of the best-response dynamics of the game. Before getting to that continuous game in the next paragraph, the main technical step of this discrete part of our construction is showing that every local equilibrium of the discrete game corresponds to a solution of the BiSperner instance we are reducing from. In order to achieve this we need to add some constraints to couple the strategies of the minimizer and the maximizer player. This step is the reason that the constraints $g(\boldsymbol{x}, \boldsymbol{y}) \leq 0$ appear in the final min-max problem that we produce.

The third and quite challenging step of the proof is to show that we can interpolate in a *smooth* and *computationally efficient* way the discrete zero-sum game of the previous step. In low dimensions (treated in Section 6) such smooth and efficient interpolation can be done in a relatively simple way using single-dimensional smooth step functions. In high dimensions, however, the smooth and efficient interpolation becomes a challenging problem and to the best of our knowledge no simple solution exists. For this reason we construct our novel *smooth and efficient interpolation coefficients* of Section 8. These are a technically involved construction that we believe will prove to be very useful for characterizing the complexity of approximate solutions of other optimization problems.

The last part of our proof is to show that all the previous steps can be implemented in an efficient way both with respect to computational but also with respect to query complexity. This part is essential for both our white-box and black-box results. Although this seems like a relatively easy step, it becomes more difficult due to the complicated expressions in our smooth and efficient interpolation coefficients used in our previous step.

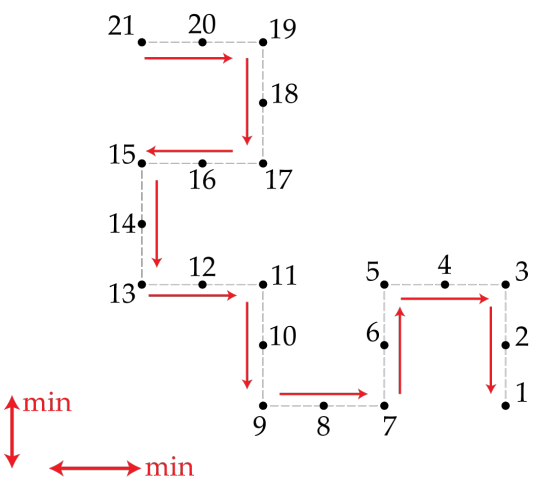
Closing this section we mention that all our NP-hardness results are proven using a cute application of Lovász Local Lemma [EL73], which provides a powerful rounding tool that can drive the inapproximability all the way up to an absolute constant.

1.2 Local Minimization vs Local Min-Max Optimization

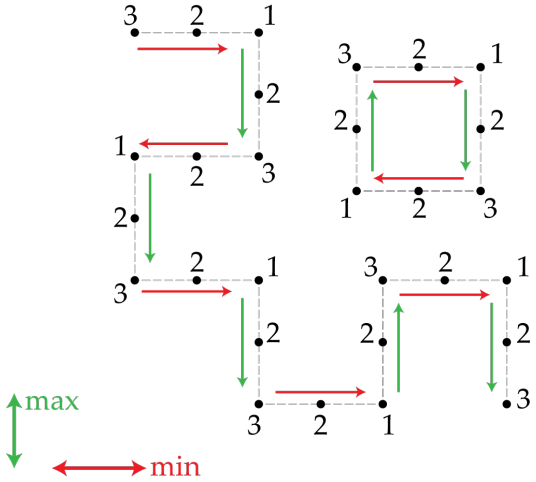
Because our proof is convoluted, involving multiple steps, it is difficult to discern from it why finding local min-max solutions is so much harder than finding local minima. For this reason, we illustrate in this section a fundamental difference between local minimization and local min-max optimization. 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.



(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.

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. For the precise step in our construction that breaks if we were to switch from a min-max to a min-min problem we refer the reader to Remark 6.9.

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 [JPY88, MP89, Pap94b] to capture the complexity of search problems that are guaranteed to have a solution. Some key complexity classes in this landscape are shown in Figure 3. We give a non-exhaustive list of intractability results for equilibrium computation: [FPT04] prove that computing pure Nash equilibria in congestion games is PLS-complete; [DGP09] and later [CDT09] show that computing approximate Nash equilib-

ria in normal-form games is PPAD-complete; [EY10] study the complexity of computing exact Nash equilibria (which may use irrational probabilities), introducing the complexity class FIXP; [VY11, CPY17] consider the complexity of computing Market equilibria; [Das13, Rub15, Rub16] consider the complexity of computing approximate Nash equilibria of constant approximation; [KM18] establish a connection between approximate Nash equilibrium computation and the SoS hierarchy; [Meh14, DFS20] study the complexity of computing Nash equilibria in specially structured games. A result that is particularly useful for our work is the result of [HPV89] which shows black-box query lower bounds for computing Brouwer fixed points of a continuous function. We use this result in Section 9 as an ingredient for proving our black-box lower bounds for computing approximate local min-max solutions.

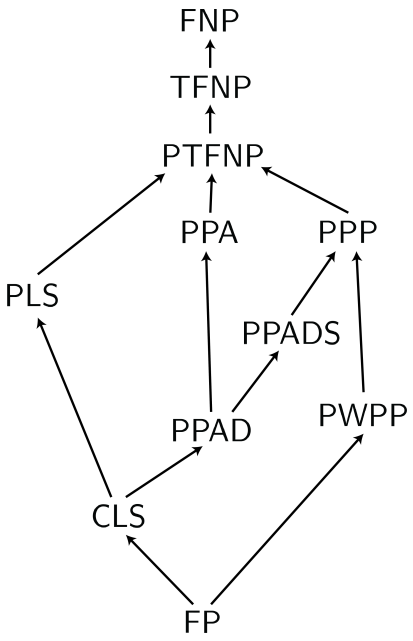


Figure 3: The complexity-theoretic landscape of total search problems in NP.

Beyond equilibrium computation and its applications to Economics and Game Theory, the study of total search problems has found profound connections to many scientific fields, including continuous optimization [DP11, DTZ18], combinatorial optimization [SY91], query complexity [BCE+95], topology [GH19], topological combinatorics and social choice theory [FG18, FG19, FRHSZ20b, FRHSZ20a], algebraic combinatorics [BIQ+17, GKSZ19], and cryptography [Jeř16, BPR15, SZZ18]. For a more extensive overview of total search problems we refer the reader to the recent survey by Daskalakis [Das18].

As already discussed, min-max optimization has intimate connections to the foundations of Game Theory, Mathematical Programming, Online Learning, Statistics, and several other fields. Recent applications of min-max optimization to Machine Learning, such as Generative Adversarial Networks and Adversarial Training, have motivated a slew of recent work targeting first-order (or other light-weight online learning) methods for solving min-max optimization problems for convex-concave, nonconvex-concave, as well as nonconvex-nonconcave objectives. Work on convex-concave and nonconvex-concave objectives has focused on obtaining online learning methods with improved rates [KM19, LJJ19, TJNO19, NSH+19, LTHC19, OX19, Zha19, ADSG19, AMLJG20, GPDO20, LJJ20] and last-iterate convergence guarantees [DISZ18, DP18, MR18, MPP18, RLLY18, HA18, ADLH19, DP19, LS19, GHP+19, MOP19, ALW19], while work on nonconvex-nonconcave problems has focused on identifying different notions of local min-max solutions [JNJ19, MV20] and studying the existence and (local) convergence properties of learning methods at these points [WZB19, MV20, MSV20].

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 $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L \|\boldsymbol{x} - \boldsymbol{y}\|_2$, 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 $\mathcal{O}_f$ such that given a point $\boldsymbol{x} \in [0,1]^d$ the oracle $\mathcal{O}_f$ 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 [NY83].
- **White Box Model.** In this model we are given the description of a polynomial-time Turing machine $\mathcal{C}_f$ 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, $\mathcal{C}_f$ 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, $\mathcal{O}_f$ or $\mathcal{C}_f$, as a *promise*, rather than enforcing these constraints in some syntactic manner.

1. **Consistency of Function Values and Gradient Values.** Given some oracle $\mathcal{O}_f$ or Turing machine $\mathcal{C}_f$, 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 $\mathcal{O}_f$ or Turing machine $\mathcal{C}_f$, 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 $\mathcal{O}_f$ or $\mathcal{C}_f$ 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 $\mathcal{O}_f$ or $\mathcal{C}_f$ 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 $\mathcal{O}_f$ or $\mathcal{C}_f$ 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.

2.1 Complexity Classes and Reductions

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

[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 [Rub16] since we agree that it describes the problem better.

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 $\boldsymbol{x}$.

> **End-of-a-Line.**
> Input: Binary circuits $\mathcal{C}_S$ (for successor) and $\mathcal{C}_P$ (for predecessor) with n inputs and n outputs.
> Output: One of the following:
> 0. $\mathbf{0}$ if either both $\mathcal{C}_P(\mathcal{C}_S(\mathbf{0}))$ and $\mathcal{C}_S(\mathcal{C}_P(\mathbf{0}))$ are equal to $\mathbf{0}$, or if they are both different than $\mathbf{0}$, where $\mathbf{0}$ is the all-0 string.
> 1. a binary string $\boldsymbol{x} \in \{0,1\}^n$ such that $\boldsymbol{x} \neq \mathbf{0}$ and $\mathcal{C}_P(\mathcal{C}_S(\boldsymbol{x})) \neq \boldsymbol{x}$ or $\mathcal{C}_S(\mathcal{C}_P(\boldsymbol{x})) \neq \boldsymbol{x}$.

To make sense of the above definition, we envision that the circuits $\mathcal{C}_S$ and $\mathcal{C}_P$ 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 $\mathcal{C}_S(\boldsymbol{x}) = \boldsymbol{y}$ and $\mathcal{C}_P(\boldsymbol{y}) = \boldsymbol{x}$. 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 $\mathcal{C}_S$ and $\mathcal{C}_P$ 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 [Pap94b].

Definition 2.4 (**PPAD**)**.** 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 [DGP09, Das18]. 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 $\mathcal{C}_M$ 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 $\|\boldsymbol{z}^\star - M(\boldsymbol{z}^\star)\|_2 < \gamma$.

While not stated exactly in this form, the following is a straightforward implication of the results presented in [CDT09].

Lemma 2.5 ([CDT09])**.** 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* $\mathcal{C}_M$ *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 $\mathcal{C}_M$ 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 $\|\nabla f(\boldsymbol{x}^\star)\|_2 < \varepsilon$.

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 $r \in \mathbb{R}_+$ we define the *closed Euclidean ball of radius r* to be the set $\mathsf{B}_d(r) = \{\boldsymbol{x} \in \mathbb{R}^d \mid \|\boldsymbol{x}\|_2 \le r\}$. We also define the *closed Euclidean ball of radius r centered at $\boldsymbol{z} \in \mathbb{R}^d$* to be the set $\mathsf{B}_d(r; \boldsymbol{z}) = \{\boldsymbol{x} \in \mathbb{R}^d \mid \|\boldsymbol{x} - \boldsymbol{z}\|_2 \le r\}$.

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 $\boldsymbol{x} \in \mathsf{B}_d(\delta; \boldsymbol{x}^\star)$.

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 $f : [0,1]^{d_1} \times [0,1]^{d_2} \to \mathbb{R}$ 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 $\boldsymbol{x} \in \mathsf{B}_{d_1}(\delta; \boldsymbol{x}^\star)$ 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 $\boldsymbol{y} \in \mathsf{B}_{d_2}(\delta; \boldsymbol{y}^\star)$ 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 $\mathcal{C}_f$ (or the interface to the oracle $\mathcal{O}_f$ 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 $\mathcal{C}_f$ 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 $\|\nabla f(\boldsymbol{x}^\star)\|_2 < \varepsilon$; 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 $\|\nabla f(\boldsymbol{x}^\star)\|_2 < \varepsilon$ 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 $\|\nabla f(\boldsymbol{x}^\star)\|_2 < \varepsilon$. 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 $\mathcal{C}_f$ 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 $\boldsymbol{x} \in \mathsf{B}_d(\delta; \boldsymbol{x}^\star) \cap \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$.

LocalMinMax.
Input: Scalars $\varepsilon, \delta, G, L, B > 0$; a polynomial-time Turing machine $\mathcal{C}_f$ 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 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 $= 1$. As we see in Section 5, 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 $\boldsymbol{x} - \nabla_x f(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{y} + \nabla_y f(\boldsymbol{x}, \boldsymbol{y})$, thus $F_{GDA}(\boldsymbol{x}, \boldsymbol{y})$ may not lie in K. We also define the "safe" version F_{sGDA}, which projects the pair $(\boldsymbol{x} - \nabla_x f(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{y} + \nabla_y f(\boldsymbol{x}, \boldsymbol{y}))$ jointly onto K. As we show in Section 5 (in particular inside the proof of Theorem 5.2), 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 $\mathcal{C}_f$ 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 $\|\boldsymbol{x}^\star - F_{GD}(\boldsymbol{x}^\star)\|_2 < \alpha$, 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 $\mathcal{C}_f$ 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 $\|(\boldsymbol{x}^\star, \boldsymbol{y}^\star) - F_{GDA}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)\|_2 < \alpha$, where $K = \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is the projection set used in the definition of F_{GDA}.

In Section 5 we show that the problems 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 Finding 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 [EL73]. The details of the proof can be found in Appendix A.

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 $O(d)$ (see Section 10).

- ▷ **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 $\boldsymbol{z}' \in \mathsf{B}_d(\delta; \boldsymbol{z})$ 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 $O(B \cdot L/\varepsilon)$ (see Appendix E). 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 $1/\varepsilon = \text{poly}(d)$, $G = \text{poly}(d)$, $L = \text{poly}(d)$, 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 $\text{poly}(G, L, 1/\varepsilon, d)$ time and function/gradient evaluations (see Appendix E).

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 $\mathcal{O}_f$ 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 $\text{poly}(d)$. Then any algorithm that has access to f only through $\mathcal{O}_f$ and computes an (ε, δ)-local min-max equilibrium has to make a number of queries to $\mathcal{O}_f$ that is exponential in at least one of the parameters: $1/\varepsilon$, G, L or d even when $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \subseteq [0, 1]^d$.*

Our main goal in the rest of the paper is to provide the proofs of Theorems 4.3, 4.4 and 4.5. In Section 5, we show how to use Brouwer's fixed point theorem to prove the existence of approximate local min-max equilibrium in the local regime. Moreover, we establish an equivalence between LocalMinMax and GDAFixedPoint, in the local regime, and show that both belong to PPAD. In Sections 6 and 7, we provide a detailed proof of our main result, i.e. Theorem 4.4. Finally, in Section 9, we show how our proof from Section 7 produces as a byproduct the black-box, unconditional lower bound of Theorem 4.5. In Section 8, we outline a useful interpolation technique which allows as to interpolate a function given its values and the values of its gradient on a hypergrid, so as to enforce the Lipschitzness and smoothness of the interpolating function. We make heavy use of this technically involved result in all our hardness proofs.

5 Existence of Approximate Local Min-Max Equilibrium

In this section, we establish the totality of LR-LocalMinMax, i.e. LocalMinMax for instances satisfying $\delta < \sqrt{2\varepsilon/L}$ as defined in Definition 4.2. In particular, we prove that every G-Lipschitz and L-smooth function admits an (ε, δ)-local min-max equilibrium, as long as $\delta < \sqrt{2\varepsilon/L}$. A byproduct of our proof is in fact that LR-LocalMinMax lies inside PPAD. Specifically the main tool that we use to prove our result is a computational equivalence between the problem of finding fixed points of the Gradient Descent/Ascent dynamic, i.e. GDAFixedPoint, and the problem LR-LocalMinMax. A similar equivalence between GDFixedPoint and LocalMin also holds, but the details of that are left to the reader as a simple exercise. Next, we first present the equivalence between GDAFixedPoint and LR-LocalMinMax, and we then show that GDAFixedPoint is in PPAD, which then also establishes that LR-LocalMinMax is in PPAD.

Theorem 5.1. *The search problems* LR-LocalMinMax *and* GDAFixedPoint *are equivalent under polynomial-time reductions. That is, there is a polynomial-time reduction from* LR-LocalMinMax *to* GDAFixedPoint *and vice versa. In particular, given some $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$ such that $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \neq \emptyset$, along with a G-Lipschitz and L-smooth function $f : \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \to \mathbb{R}$:*

1. *For arbitrary $\varepsilon > 0$ and $0 < \delta < \sqrt{2\varepsilon/L}$, suppose that $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ is an α-approximate fixed point of F_{GDA}, i.e., $\|(\boldsymbol{x}^*, \boldsymbol{y}^*) - F_{GDA}(\boldsymbol{x}^*, \boldsymbol{y}^*)\|_2 < \alpha$, where $\alpha \leq \frac{\sqrt{(G+\delta)^2 + 4(\varepsilon - \frac{L}{2}\delta^2)} - (G+\delta)}{2}$. Then $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is also a (ε, δ)-local min-max equilibrium of f.*

2. *For arbitary $\alpha > 0$, suppose that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an (ε, δ)-local min-max equilibrium of f for $\varepsilon = \frac{\alpha^2 \cdot L}{(5L+2)^2}$ and $\delta = \sqrt{\varepsilon / L}$. Then $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is also an α-approximate fixed point of F_{GDA}.*

The proof of Theorem 5.1 is presented in Appendix B.1. As already discussed, we use GDAFixed-Point as an intermediate step to establish the totality of LR-LocalMinMax and to show its inclusion in PPAD. This leads to the following theorem.

Theorem 5.2. *The computational problems* GDAFixedPoint *and* LR-LocalMinMax *are both total search problems and they both lie in* PPAD.

Observe that Theorem 4.3 is implied by Theorem 5.2 whose proof is presented in Appendix B.2.

6 Hardness of Local Min-Max Equilibrium – Four-Dimensions

In Section 5, we established that LR-LocalMinMax belongs to PPAD. Our proof is via the intermediate problem GDAFixedPoint which we showed that it is computationally equivalent to LR-LocalMinMax. Our next step is to prove the PPAD-hardness of LR-LocalMinMax using again GDAFixedPoint as an intermediate problem.

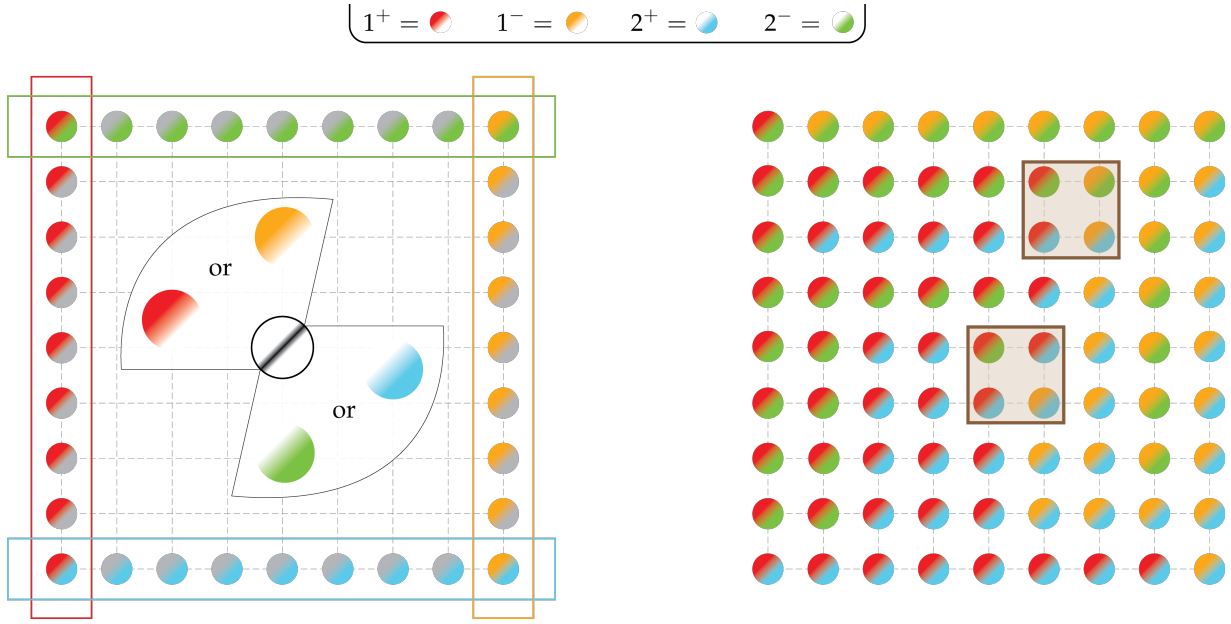
In this section we prove that GDAFixedPoint is PPAD-hard in four dimensions. To establish this hardness result we introduce a variant of the classical 2D-Sperner problem which we call 2D-BiSperner which we show is PPAD-hard. The main technical part of our proof is to show that 2D-BiSperner with input size n reduces to GDAFixedPoint, with input size $\text{poly}(n)$, $\alpha = \exp(-\text{poly}(n))$, $G = L = \exp(\text{poly}(n))$, and $B = 2$. This reduction proves the hardness of GDAFixedPoint. Formally, our main result of this section is the following theorem.

Theorem 6.1. *The problem* GDAFixedPoint *is* PPAD*-complete even in dimension $d = 4$ and $B = 2$. Therefore,* LR-LocalMinMax *is* PPAD*-complete even in dimension $d = 4$ and $B = 2$.*

The above result excludes the existence of an algorithm for GDAFixedPoint whose running time is $\text{poly}(\log G, \log L, \log(1/\alpha), B)$ and, equivalently, the existence of an algorithm for the problem LR-LocalMinMax with running time $\text{poly}(\log G, \log L, \log(1/\varepsilon), \log(1/\delta), B)$, unless FP = PPAD. Observe that it would not be possible to get a stronger hardness result for the four dimensional GDAFixedPoint problem since it is simple to construct brute-force search algorithms with running time $\text{poly}(1/\alpha, G, L, B)$. We elaborate more on such algorithms towards the end of this section. In order to prove the hardness of GDAFixedPoint for polynomially (rather than exponentially) bounded (in the size of the input) values of $1/\alpha$, G, and L (See Theorem 4.4) we need to consider optimization problems in higher dimensions. This is the problem that we explore in Section 7. Beyond establishing the hardness of the problem for $d = 4$ dimensions, the purpose of this section is to provide a simpler reduction that helps in the understanding of our main result in the next section.

6.1 The 2D Bi-Sperner Problem

We start by introducing the 2D-BiSperner problem. Consider a coloring of the $N \times N$, 2-dimensional grid, where instead of coloring each vertex of the grid with a single color (as in Sperner's lemma), each vertex is colored via a combination of two out of four available colors. The four available colors are $1^-, 1^+, 2^-, 2^+$. The five rules that define a proper coloring of the $N \times N$ grid are the following.



Summary of the rules

Example of proper coloring

Figure 4: *Left*: Summary of the rules from a proper coloring of the grid. The gray color on the left and the right side can be replaced with either blue or green. Similarly the gray color on the top and the bottom side can be replaced with either red or yellow. *Right:* An example of a proper coloring of a 9×9 grid. The brown boxes indicate the two panchromatic cells, i.e., the cells where all the four available colors appear.

1. The first color of every vertex is either 1^- or 1^+ and the second color is either 2^- or 2^+.
2. The first color of all vertices on the left boundary of the grid is 1^+.
3. The first color of all vertices on the right boundary of the grid is 1^-.
4. The second color of all vertices on the bottom boundary of the grid is 2^+.
5. The second color of all vertices on the top boundary of the grid is 2^-.

Using similar proof ideas as in Sperner's lemma it is not hard to establish via a combinatorial argument that, in every proper coloring of the $N \times N$ grid, there exists a square cell where each of the four colors in $\{1^-, 1^+, 2^-, 2^+\}$ appears in at least one of its vertices. We call such a cell a *panchromatic square*. In the 2D-BiSperner problem, defined formally below, we are given the description of some coloring of the grid and are asked to find either a panchromatic square or the violation of the proper coloring conditions. In this paper, we will not present a direct combinatorial argument guaranteeing the existence of panchromatic squares under proper colorings of the grid, since the existence of panchromatic squares will be implied by the totality of the 2D-BiSperner problem, which will follow from our reduction from 2D-BiSperner to GDAFixedPoint as well as our proofs in Section 5 establishing the totality of GDAFixedPoint. In Figure 4 we summarize the five rules that define proper colorings and we present an example of a proper coloring of the grid with 9 discrete points on each side.

In order to formally define the computational problem 2D-BiSperner in a way that is useful for our reductions we need to allow for colorings of the $N \times N$ grid described in a succinct way, where the value N can be exponentially large compared to the size of the input to the problem. A standard way to do this, introduced by [Pap94b] in defining the computational version of Sperner's lemma, is to describe a coloring via a binary circuit $\mathcal{C}_l$ that takes as input the coordinates of a vertex in the grid and outputs the combination of colors that is used to color this vertex. In the input, each one of the two coordinates of the input vertex is given via the binary representation of a number in $[N]-1$. Setting $N = 2^n$ we have that the representation of each coordinate belongs to $\{0,1\}^n$. In the rest of the section we abuse the notation and we use a coordinate $i \in \{0,1\}^n$ both as a binary string and as a number in $[2^n]-1$ and it is clear from the context which of the two we use. The output of $\mathcal{C}_l$ should be a combination of one of the colors $\{1^-, 1^+\}$ and one of the colors $\{2^-, 2^+\}$. We represent this combination as a pair of $\{-1,1\}^2$. The first coordinate of this pair refers to the choice of 1^- or 1^+ and the second coordinate refers to the choice of 2^- or 2^+.

In the definition of the computational problem 2D-BiSperner the input is a circuit $\mathcal{C}_l$, as described above. One type of possible solutions to 2D-BiSperner is providing a pair of coordinates $(i,j) \in \{0,1\}^n \times \{0,1\}^n$ indexing a cell of the grid whose bottom left vertex is (i,j). For this type of solution to be valid it must be that the output of $\mathcal{C}_l$ when evaluated on all the vertices of this square contains at least one negative and one positive value for each one of the two output coordinates of $\mathcal{C}_l$, i.e. the cell must be panchromatic. Another type of possible solution to 2D-BiSperner is a vertex whose coloring violates the proper coloring conditions for the boundary, namely 2–5 above. For notational convenience we refer to the first coordinate of the output of $\mathcal{C}_l$ by $\mathcal{C}_l^1$ and to the second coordinate by $\mathcal{C}_l^2$. The formal definition of the computational problem 2D-BiSperner is then the following.

2D-BiSperner.
Input: A boolean circuit $\mathcal{C}_l : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}^2$.
Output: A vertex $(i,j) \in \{0,1\}^n \times \{0,1\}^n$ such that one of the following holds

1. $i \neq \mathbf{1}$, $j \neq \mathbf{1}$, and
$$\bigcup_{\substack{i'-i\in\{0,1\}\\ j'-j\in\{0,1\}}} \mathcal{C}_l^1(i',j') = \{-1,1\} \quad \text{and} \quad \bigcup_{\substack{i'-i\in\{0,1\}\\ j'-j\in\{0,1\}}} \mathcal{C}_l^2(i',j') = \{-1,1\}, \text{ or}$$
2. $i = \mathbf{0}$ and $\mathcal{C}_l^1(i,j) = -1$, or
3. $i = \mathbf{1}$ and $\mathcal{C}_l^1(i,j) = +1$, or
4. $j = \mathbf{0}$ and $\mathcal{C}_l^2(i,j) = -1$, or
5. $j = \mathbf{1}$ and $\mathcal{C}_l^2(i,j) = +1$.

Our next step is to show that the problem 2D-BiSperner is PPAD-hard. Thus our reduction from 2D-BiSperner to GDAFixedPoint in the next section establishes both the PPAD-hardness of GDAFixedPoint and the inclusion of 2D-BiSperner to PPAD.

Lemma 6.2. *The problem* 2D-BiSperner *is* PPAD*-hard.*

Proof. To prove this Lemma we will use Lemma 2.5. Let $\mathcal{C}_M$ be a polynomial-time Turing machine that computes a function $M : [0,1]^2 \to [0,1]^2$ that is L-Lipschitz. We know from Lemma 2.5 that

finding γ-approximate fixed points of M is PPAD-hard. We will use $\mathcal{C}_M$ to define a circuit $\mathcal{C}_l$ such that a solution of 2D-BiSperner with input $\mathcal{C}_l$ will give us a γ-approximate fixed point of M.

Consider the function $g(\boldsymbol{x}) = M(\boldsymbol{x}) - \boldsymbol{x}$. Since M is L-Lipschitz, the function $g : [0,1]^2 \to [-1,1]^2$ is also $(L+1)$-Lipschitz. Additionally g can be easily computed via a polynomial-time Turing machine $\mathcal{C}_g$ that uses $\mathcal{C}_M$ as a subroutine. We construct a proper coloring of a fine grid of $[0,1]^2$ using the signs of the outputs of g. Namely we set $n = \lceil \log(L/\gamma) + 2 \rceil$ and this defines a $2^n \times 2^n$ grid over $[0,1]^2$ that is indexed by $\{0,1\}^n \times \{0,1\}^n$. Let $g_\eta : [0,1]^2 \to [-1,1]^2$ be the function that the Turing Machine $\mathcal{C}_g$ evaluate when the requested accuracy is $\eta > 0$. Now we can define the circuit $\mathcal{C}_l$ as follows, [5]

$$\mathcal{C}_l^1(i,j) = \begin{cases} 1 & i = 0 \\ -1 & i = 2^n - 1 \\ 1 & g_{\eta,1}\left(\frac{i}{2^n-1}, \frac{j}{2^n-1}\right) \geq 0 \text{ and } i \neq -1\,' \\ -1 & g_{\eta,1}\left(\frac{i}{2^n-1}, \frac{j}{2^n-1}\right) < 0 \text{ and } i \neq 0 \end{cases}$$

$$\mathcal{C}_l^2(i,j) = \begin{cases} 1 & i = 0 \\ -1 & i = 2^n - 1 \\ 1 & g_{\eta,2}\left(\frac{i}{2^n-2}, \frac{j}{2^n-1}\right) \geq 0 \text{ and } i \neq -1\,' \\ -1 & g_{\eta,2}\left(\frac{i}{2^n-2}, \frac{j}{2^n-1}\right) < 0 \text{ and } i \neq 0 \end{cases}$$

where g_i is the ith output coordinate of g. It is not hard then to observe that the coloring $\mathcal{C}_l$ is proper, i.e. it satisfies the boundary conditions due to the fact that the image of M is always inside $[0,1]^2$. Therefore the only possible solution to 2D-BiSperner with input $\mathcal{C}_l$ is a cell that contains all the colors $\{1^-, 1^+, 2^-, 2^+\}$. Let (i,j) be the bottom-left vertex of this cell which we denote by R, namely

$$R = \left\{ \boldsymbol{x} \in [0,1]^2 \mid x_1 \in \left[\frac{i}{2^n - 1}, \frac{i+1}{2^n - 1} \right], x_2 \in \left[\frac{j}{2^n - 1}, \frac{j+1}{2^n - 1} \right] \right\}.$$

Claim 6.3. *Let $\eta = \frac{\gamma}{2\sqrt{2}}$, there exists $\boldsymbol{x} \in R$ such that $|g_1(\boldsymbol{x})| \leq \frac{\gamma}{2\sqrt{2}}$ and $\boldsymbol{y} \in R$ such that $|g_2(\boldsymbol{y})| \leq \frac{\gamma}{2\sqrt{2}}$.*

Proof of Claim 6.3. We will prove the existence of $\boldsymbol{x}$ and the existence of $\boldsymbol{y}$ follows using an identical argument. If there exists a corner $\boldsymbol{x}$ of R such that $g_1(\boldsymbol{x})$ is in the range $[-\eta, \eta]$ then the claim follows. Suppose not. Using this together with the fact that the first color of one of the corners of R is 1^- and also the first color of one of the corners of R is 1^+ we conclude that there exist points $\boldsymbol{x}, \boldsymbol{x}'$ such that $g_{\eta,1}(\boldsymbol{x}) \geq 0$ and $g_{\eta,1}(\boldsymbol{x}') \leq 0$ [6]. But we have that $\left\| g_\eta - g \right\|_2 \leq \eta$. This together with the fact that $g_1(\boldsymbol{x}) \notin [-\eta, \eta]$ and $g_1(\boldsymbol{x}') \notin [-\eta, \eta]$ implies that $g_1(\boldsymbol{x}) \geq 0$ and also $g_1(\boldsymbol{x}') \leq 0$. But because of the L-Lipschitzness of g and because the distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ is at most $\sqrt{2}\frac{\gamma}{4L}$ we conclude that $|g_1(\boldsymbol{x}) - g_1(\boldsymbol{x}')| \leq \frac{\gamma}{2\sqrt{2}}$. Hence due to the signs of $g_1(\boldsymbol{x})$ and $g_1(\boldsymbol{x}')$ we conclude that $|g_1(\boldsymbol{x})| \leq \frac{\gamma}{2\sqrt{2}}$. The same way we can prove that $|g_1(\boldsymbol{y})| \leq \frac{\gamma}{2\sqrt{2}}$ and the claim follows. □

[5] We remind that we abuse the notation and we use a coordinate $i \in \{0,1\}^n$ both as a binary string and as a number in $([2^n - 1] - 1)$ and it is clear from the context which of the two we use.

[6] The latter is inaccurate for the cases where the vertex $(0,j)$ belongs to either facets $i = 0$ or $i = 2^n - 1$. Notice that the coloring in such vertices does not depend on the value of g_η. However in case where the color of such a corner is not consistent with the value of g_η, i.e. $g_{\eta,1}(0,j) < 0$ and $\mathcal{C}_l^1(0,j) = 1$ then this means that $|g_1(0,j)| \leq \eta$. This is due to the fact that $g_1(0,j) \geq 0$ and $|g_1(0,j) - g_{1,\eta}(0,j)| \leq \eta$.

Using the Claim 6.3 and the L-Lipschitzness of g we get that for every $\boldsymbol{z} \in R$

$$|g_1(\boldsymbol{z}) - g_1(\boldsymbol{x})| \leq L \|\boldsymbol{x} - \boldsymbol{z}\|_2 \leq \sqrt{2} \cdot L \cdot \frac{\gamma}{4L} \implies |g_1(\boldsymbol{z})| \leq \frac{\gamma}{\sqrt{2}}, \text{ and}$$
$$|g_2(\boldsymbol{z}) - g_2(\boldsymbol{y})| \leq L \|\boldsymbol{y} - \boldsymbol{z}\|_2 \leq \sqrt{2} \cdot L \cdot \frac{\gamma}{4L} \implies |g_2(\boldsymbol{z})| \leq \frac{\gamma}{\sqrt{2}}$$

where we have used also the fact that for any two points $\boldsymbol{z}, \boldsymbol{w}$ it holds that $\|\boldsymbol{z} - \boldsymbol{w}\|_2 \leq \sqrt{2}\frac{\gamma}{4L}$ which follows from the definition of the size of the grid. Therefore we have that $\|g(\boldsymbol{z})\|_2 \leq \gamma$ and hence $\|M(\boldsymbol{z}) - \boldsymbol{z}\|_2 \leq \gamma$ which implies that any point $\boldsymbol{z} \in R$ is a γ-approximate fixed point of M and the lemma follows. □

Now that we have established the PPAD-hardness of 2D-BiSperner we are ready to present our main result of this section which is a reduction from 2D-BiSperner to GDAFixedPoint.

6.2 From 2D Bi-Sperner to Fixed Points of Gradient Descent/Ascent

We start with presenting a construction of a Lipschitz and smooth real-valued function $f : [0,1]^2 \times [0,1]^2 \to \mathbb{R}$ based on a given coloring circuit $\mathcal{C}_l : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}^2$. Then in Section 6.2.1 we will show that any solution to GDAFixedPoint with input the representation $\mathcal{C}_f$ of f is also a solution to the 2D-BiSperner problem with input $\mathcal{C}_l$. Constructing Lipschitz and smooth functions based on only local information is a surprisingly challenging task in high-dimensions as we will explain in detail in Section 7. Fortunately in the low-dimensional case that we consider in this section the construction is much more simple and the main ideas of our reduction are more clear.

The basic idea of the construction of f consists in interpreting the coloring of a given point in the grid as the directions of the gradient of $f(\boldsymbol{x}, \boldsymbol{y})$ with respect to the variables x_1, y_1 and x_2, y_2 respectively. More precisely, following the ideas in the proof of Lemma 6.2, we divide the $[0,1]^2$ square in *square-cells* of length $1/(N-1) = 1/(2^n - 1)$ where the corners of these cells correspond to vertices of the $N \times N$ grid of the 2D-BiSperner instance described by $\mathcal{C}_l$. When $\boldsymbol{x}$ is on a vertex of this grid, the first color of this vertex determines the direction of gradient with respect to the variables x_1 and y_1, while the second color of this vertex determines the direction of the gradient of the variables x_2 and y_2. As an example, if $\boldsymbol{x} = (x_1, x_2)$ is on a vertex of the $N \times N$ grid, and the coloring of this vertex is $(1^-, 2^+)$, i.e. the output of $\mathcal{C}_l$ on this vertex is $(-1, +1)$, then we would like to have

$$\frac{\partial f}{\partial x_1}(\boldsymbol{x}, \boldsymbol{y}) \geq 0, \quad \frac{\partial f}{\partial y_1}(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \quad \frac{\partial f}{\partial x_2}(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \quad \frac{\partial f}{\partial y_2}(\boldsymbol{x}, \boldsymbol{y}) \geq 0.$$

The simplest way to achieve this is to define the function f locally close to $(\boldsymbol{x}, \boldsymbol{y})$ to be equal to

$$f(\boldsymbol{x}, \boldsymbol{y}) = (x_1 - y_1) - (x_2 - y_2).$$

Similarly, if $\boldsymbol{x}$ is on a vertex of the $N \times N$ grid, and the coloring of this vertex is $(1^-, 2^-)$, i.e. the output of $\mathcal{C}_l$ on this vertex is $(-1, -1)$, then we would like to have

$$\frac{\partial f}{\partial x_1}(\boldsymbol{x}, \boldsymbol{y}) \geq 0, \quad \frac{\partial f}{\partial y_1}(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \quad \frac{\partial f}{\partial x_2}(\boldsymbol{x}, \boldsymbol{y}) \geq 0, \quad \frac{\partial f}{\partial y_2}(\boldsymbol{x}, \boldsymbol{y}) \leq 0.$$

The simplest way to achieve this is to define the function f locally close to $(\boldsymbol{x}, \boldsymbol{y})$ to be equal to

$$f(\boldsymbol{x}, \boldsymbol{y}) = (x_1 - y_1) + (x_2 - y_2).$$

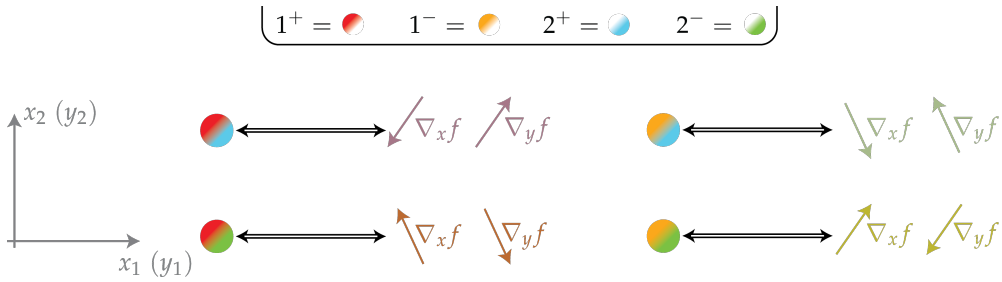


Figure 5: The correspondence of the colors of the vertices of the $N \times N$ grid with the directions of the gradient of the function f that we design.

In Figure 5 we show pictorially the correspondence of the colors of the vertices of the grid with the gradient of the function f that we design. As shown in the figure, any set of vertices that share at least one of the colors $1^+, 1^-, 2^+, 2^-$, agree on the direction of the gradient with respect the horizontal or the vertical axis. This observation is one of the main ingredients in the proof of correctness of our reduction that we present later in this section.

When $\boldsymbol{x}$ is not on a vertex of the $N \times N$ grid then our goal is to define f via interpolating the functions corresponding to the corners of the cell in which $\boldsymbol{x}$ belongs. The reason that this interpolation is challenging is that we need to make sure the following properties are satisfied

- ▷ the resulting function f is both Lipschitz and smooth inside every cell,
- ▷ the resulting function f is both Lipschitz and smooth even at the boundaries of every cell, where two differect cells stick together,
- ▷ no solution to the GDAFixedPoint problem is created inside cells that are not solutions to the 2D-BiSperner problem. In particular, it has to be true that if all the vertices of one cell agree on some color then the gradient of f inside that cell has large enough gradient in the corresponding direction.

For the low dimensional case, that we explore in this section, satisfying the first two properties is not a very difficult task, whereas for the third property we need to be careful and achieving this property is the main technical contribution of this section. On the contrary, for the high-dimensional case that we explore in Section 7 even achieving the first two properties is very challenging and technical.

As we will see in Section 6.2.1, if we accomplish a construction of a function f with the aforementioned properties, then the fixed points of the projected Gradient Descent/Ascent can only appear inside cells that have all of the colors $\{1^-, 1^+, 2^-, 2^+\}$ at their corners. To see this consider a cell that misses some color, e.g. 1^+. Then all the corners of this cell have as first color 1^-. Since f is defined as interpolation of the functions in the corners of the cells, with the aforementioned properties, inside that cell there is always a direction with respect to x_1 and y_1 for which the gradient is large enough. Hence any point inside that cell cannot be a fixed point of the projected Gradient Descent/Ascent. Of course this example provides just an intuition of our construction and ignores case where the cell is on the boundary of the grid. We provide a detailed explanation of this case in Section 6.2.1.

The above neat idea needs some technical adjustments in order to work. At first, the interpolation of the function in the interior of the cell must be smooth enough so that the resulting

function is both Lipschitz and smooth. In order to satisfy this, we need to choose appropriate coefficients of the interpolation that interpolate smoothly not only the value of the function but also its derivatives. For this purpose we use the following smooth step function of order 1.

Definition 6.4 (Smooth Step Function of Order 1)**.** We define $S_1 : [0,1] \to [0,1]$ to be the *smooth step function of order* 1 that is equal to $S_1(x) = 3x^2 - 2x^3$. Observe that the following hold $S_1(0) = 0$, $S_1(1) = 1$, $S_1'(0) = 0$, and $S_1'(1) = 0$.

As we have discussed, another issue is that since the interpolation coefficients depend on the value of $\boldsymbol{x}$ it could be that the derivatives of these coefficients overpower the derivatives of the functions that we interpolate. In this case we could be potentially creating fixed points of Gradient Descent/Ascent even in *non* panchromatic squares. As we will see later the magnitude of the derivatives from the interpolation coefficients depends on the differences $x_1 - y_1$ and $x_2 - y_2$. Hence if we ensure that these differences are small then the derivatives of the interpolation coefficients will have to remain small and hence they can never overpower the derivatives from the corners of every cell. This is the place in our reduction where we add the constraints $\boldsymbol{A} \cdot (\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{b}$ that define the domain of the function f as we describe in Section 3.

Now that we have summarized the main ideas of our construction we are ready for the formal definition of f based on the coloring circuit $\mathcal{C}_l$.

Definition 6.5 (Continuous and Smooth Function from Colorings of 2D-Bi-Sperner)**.** Given a binary circuit $\mathcal{C}_l : \{0,1\}^n \times \{0,1\}^n \to \{-1,1\}^2$, we define the function $f_{\mathcal{C}_l} : [0,1]^2 \times [0,1]^2 \to \mathbb{R}$ as follows. For any $\boldsymbol{x} \in [0,1]^2$, let $A = (i_A, j_A)$, $B = (i_B, j_B)$, $C = (i_C, j_C)$, $D = (i_D, j_D)$ be the vertices of the cell of the $N(= 2^n) \times N$ grid which contains $\boldsymbol{x}$ and $\boldsymbol{x}^A$, $\boldsymbol{x}^B$, $\boldsymbol{x}^C$ and $\boldsymbol{x}^C$ the corresponding points in the unit square $[0,1]^2$, i.e. $x_1^A = i_A/(2^n - 1)$, $x_2^A = j_A/(2^n - 1)$ etc. Let also A be down-left corner of this cell and B, C, D be the rest of the vertices in clockwise order, then we define

$$f_{\mathcal{C}_l}(\boldsymbol{x}, \boldsymbol{y}) = \alpha_1(\boldsymbol{x}) \cdot (y_1 - x_1) + \alpha_2(\boldsymbol{x}) \cdot (y_2 - x_2)$$

where the coefficients $\alpha_1(\boldsymbol{x}), \alpha_2(\boldsymbol{x}) \in [-1,1]$ are defined as follows

$$\begin{aligned} \alpha_i(\boldsymbol{x}) \quad = \quad & S_1\left(\frac{x_1^C - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2^C - x_2}{\delta}\right) \cdot \mathcal{C}_l^i(A) + S_1\left(\frac{x_1^D - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^D}{\delta}\right) \cdot \mathcal{C}_l^i(B) \\ & + S_1\left(\frac{x_1 - x_1^A}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^A}{\delta}\right) \cdot \mathcal{C}_l^i(C) + S_1\left(\frac{x_1 - x_1^B}{\delta}\right) \cdot S_1\left(\frac{x_2^B - x_2}{\delta}\right) \cdot \mathcal{C}_l^i(D) \end{aligned}$$

where $\delta \triangleq 1/(N-1) = 1/(2^n - 1)$.

In Figure 6 we present an example of the application of Definition 6.5 to a specific cell with some given coloring on the corners.

An important property of the definition of the function $f_{\mathcal{C}_l}$ is that the coefficients used in the definition of α_i have the following two properties

$$S_1\left(\frac{x_1^C - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2^C - x_2}{\delta}\right) \geq 0, \; S_1\left(\frac{x_1^D - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^D}{\delta}\right) \geq 0,$$

$$S_1\left(\frac{x_1 - x_1^A}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^A}{\delta}\right) \geq 0, \; S_1\left(\frac{x_1 - x_1^B}{\delta}\right) \cdot S_1\left(\frac{x_2^B - x_2}{\delta}\right) \geq 0, \text{ and}$$

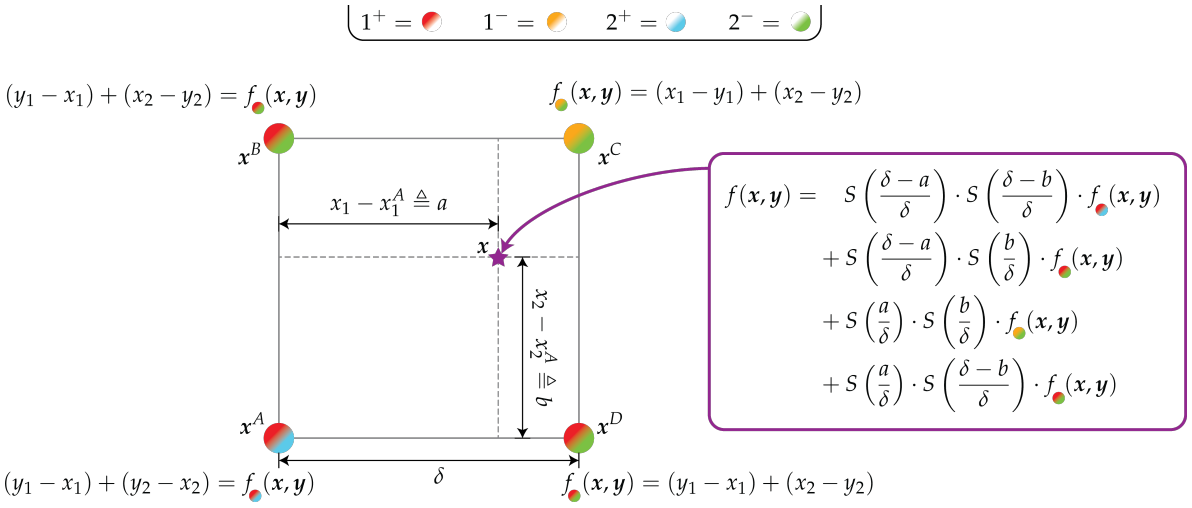


Figure 6: Example of the definition of the Lipschitz and smooth function f on some cell given the coloring on the corners of the cell. For details see Definition 6.5.

$$S_1\left(\frac{x_1^C - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2^C - x_2}{\delta}\right) + S_1\left(\frac{x_1^D - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^D}{\delta}\right)$$
$$+ S_1\left(\frac{x_1 - x_1^A}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^A}{\delta}\right) + S_1\left(\frac{x_1 - x_1^B}{\delta}\right) \cdot S_1\left(\frac{x_2^B - x_2}{\delta}\right) = 1.$$

Hence the function $f_{\mathcal{C}_l}$ inside a cell is a smooth convex combination of the functions on the corners of the cell, as is suggested from Figure 6. Of course there are many ways to define such convex combination but in our case we use the smooth step function S_1 to ensure the Lipschitz continuous gradient of the overall function $f_{\mathcal{C}_l}$. We prove this formally in the next lemma.

Lemma 6.6. *Let $f_{\mathcal{C}_l}$ be the function defined based on a coloring circuit $\mathcal{C}_l$, as per Definition 6.5. Then $f_{\mathcal{C}_l}$ is continuous and differentiable at any point $(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]^4$. Moreover, $f_{\mathcal{C}_l}$ is $\Theta(1/\delta)$-Lipschitz and $\Theta(1/\delta^2)$-smooth in the whole 4-dimensional hypercube $[0,1]^4$, where $\delta = 1/(N-1) = 1/(2^n - 1)$.*

Proof. Clearly from Definition 6.5, $f_{\mathcal{C}_l}$ is differentiable at any point $(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]^4$ in which $\boldsymbol{x}$ lies on the strict interior of its respective cell. In this case the derivative with respect to x_1 is

$$\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}, \boldsymbol{y})}{\partial x_1} = \frac{\partial \alpha_1(\boldsymbol{x})}{\partial x_1} \cdot (y_1 - x_1) - \alpha_1(\boldsymbol{x}) + \frac{\partial \alpha_2(\boldsymbol{x})}{\partial x_1} \cdot (y_2 - x_2).$$

where for $\partial \alpha_1(\boldsymbol{x}) / \partial x_1$ we have that

$$\frac{\partial \alpha_1(\boldsymbol{x})}{\partial x_1} = -\frac{1}{\delta} S_1'\left(\frac{x_1^C - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2^C - x_2}{\delta}\right) \cdot \mathcal{C}_l^1(A)$$
$$- \frac{1}{\delta} S_1'\left(\frac{x_1^D - x_1}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^D}{\delta}\right) \cdot \mathcal{C}_l^1(B)$$
$$+ \frac{1}{\delta} S_1'\left(\frac{x_1 - x_1^A}{\delta}\right) \cdot S_1\left(\frac{x_2 - x_2^A}{\delta}\right) \cdot \mathcal{C}_l^1(C)$$
$$+ \frac{1}{\delta} S_1'\left(\frac{x_1 - x_1^B}{\delta}\right) \cdot S_1\left(\frac{x_2^B - x_2}{\delta}\right) \cdot \mathcal{C}_l^1(D).$$

Now since $\max_{z\in[0,1]} |S_1'(z)| \leq 6$, we can conclude that $\left|\frac{\partial \alpha_1(\boldsymbol{x})}{\partial x_1}\right| \leq 24/\delta$. Similarly we can prove that $\left|\frac{\partial \alpha_2(\boldsymbol{x})}{\partial x_1}\right| \leq 24/\delta$, which combined with $|\alpha_1(\boldsymbol{x})| \leq 1$ implies $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial x_1}\right| \leq O(1/\delta)$. Using similar reasoning we can prove that $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial x_2}\right| \leq O(1/\delta)$ and that $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial y_i}\right| \leq 1$ for $i = 1, 2$. Hence

$$\left\|\nabla f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})\right\|_2 \leq O(1/\delta).$$

The only thing we are missing to prove the Lipschitzness of $f_{\mathcal{C}_l}$ is to prove its continuity on the boundaries of the cells of our subdivision. Suppose $\boldsymbol{x}$ lies on the boundary of some cell, e.g. let $\boldsymbol{x}$ lie on edge (C, D) of one cell that is the same as the edge (A', B') of the cell to the right of that cell. Since $S_1(0) = 0$, $S_1'(0) = 0$ and $S_1'(1) = 0$ it holds that $\partial\alpha_1(\boldsymbol{x})/\partial x_1 = 0$ and the same for α_2. Therefore the value of $\partial f_{\mathcal{C}_l}/\partial x_1$ remains the same no matter the cell according to which it was calculated. As a result, $f_{\mathcal{C}_l}$ is differentiable with respect to x_1 even if $\boldsymbol{x}$ belongs in the boundary of its cell. Using the exact same reasoning for the rest of the variables, one can show that the function $f_{\mathcal{C}_l}$ is differentiable at any point $(\boldsymbol{x},\boldsymbol{y}) \in [0,1]^4$ and because of the aforementioned bound on the gradient $\nabla f_{\mathcal{C}_l}$ we can conclude that $f_{\mathcal{C}_l}$ is $O(1/\delta)$-Lipschitz.

Using very similar calculations, we can compute the closed formulas of the second derivatives of $f_{\mathcal{C}_l}$ and using the bounds $\left|f_{\mathcal{C}_l}(\cdot)\right| \leq 2$, $|S_1(\cdot)| \leq 1$, $|S_1'(\cdot)| \leq 6$, and $|S_1''(\cdot)| \leq 6$, we can prove that each entry of the Hessian $\nabla^2 f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})$ is bounded by $O(1/\delta^2)$ and thus

$$\left\|\nabla^2 f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})\right\|_2 \leq O(1/\delta^2)$$

which implies the $\Theta(1/\delta^2)$-smoothness of $f_{\mathcal{C}_l}$. □

6.2.1 Description and Correctness of the Reduction – Proof of Theorem 6.1

In this section, we present and prove the exact polynomial-time construction of the instance of the problem GDAFixedPoint from an instance $\mathcal{C}_l$ of the problem 2D-BiSperner.

(+) Construction of Instance for Fixed Points of Gradient Descent/Ascent.
Our construction can be described via the following properties.

- The payoff function is the real-valued function $f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})$ from the Definition 6.5.
- The domain is the polytope $\mathcal{P}(\boldsymbol{A},\boldsymbol{b})$ that we described in Section 3. The matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ have constant size and they are computed so that the following inequalities hold

$$x_1 - y_1 \leq \Delta, \quad y_1 - x_1 \leq \Delta, \quad x_2 - y_2 \leq \Delta, \quad \text{and} \quad y_2 - x_2 \leq \Delta \tag{6.1}$$

  where $\Delta = \delta/12$ and $\delta = 1/(N-1) = 1/(2^n - 1)$.
- The parameter α is set to be equal to $\Delta/3$.
- The parameters G and L are set to be equal to the upper bounds on the Lipschitzness and the smoothness of $f_{\mathcal{C}_l}$ respectively that we derived in Lemma 6.6. Namely we have that $G = O(1/\delta) = O(2^n)$ and $L = O(1/\delta^2) = O(2^{2n})$.

The first thing that is simple to observe in the above reduction is that it runs in polynomial time with respect to the size of the the circuit $\mathcal{C}_l$ which is the input to the 2D-BiSperner problem that we started with. To see this, recall from the definition of GDAFixedPoint that our reduction

needs to output: (1) a Turing machine $\mathcal{C}_{f_{\mathcal{C}_l}}$ that computes the value and the gradient of the function $f_{\mathcal{C}_l}$ in time polynomial in the number of requested bits of accuracy; (2) the required scalars α, G, and L. For the first, we observe from the definition of $f_{\mathcal{C}_l}$ that it is actually a piecewise polynomial function with a closed form that only depends on the values of the circuit $\mathcal{C}_l$ on the corners of the corresponding cell. Since the size of $\mathcal{C}_l$ is the size of the input to 2D-BiSperner we can easily construct a polynomial-time Turing machine that computes both function value and the gradient of the piecewise polynomial function $f_{\mathcal{C}_l}$. Also, from the aforementioned description of the reduction we have that $\log(G)$, $\log(L)$ and $\log(1/\alpha)$ are linear in n and hence we can construct the binary representation of all this scalars in time $O(n)$. The same is true for the coefficients of $\boldsymbol{A}$ and $\boldsymbol{b}$ as we can see from their definition in (+). Hence we conclude that our reduction runs in time that is polynomial in the size of the circuit $\mathcal{C}_l$.

The next thing to observe is that, according to Lemma 6.6, the function $f_{\mathcal{C}_l}$ is both G-Lipschitz and L-smooth and hence the output of our reduction is a valid input for the promise problem GDAFixedPoint. So the last step to complete the proof of Theorem 6.1 is to prove that the vector $\boldsymbol{x}^\star$ of every solution $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ of GDAFixedPoint with input $\mathcal{C}_{f_{\mathcal{C}_l}}$, lies in a cell that is either panchromatic or violates the rules for proper coloring, in any of these cases we can find a solution to the 2D-BiSperner problem. This proves that our construction reduces 2D-BiSperner to GDAFixedPoint.

We prove this last statement in Lemma 6.8, but before that we need the following technical lemma that will be useful to argue about solution on the boundary of $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$.

Lemma 6.7. *Let $\mathcal{C}_l$ be an input to the* 2D-BiSperner *problem, let $f_{\mathcal{C}_l}$ be the corresponding G-Lipschitz and L-smooth function defined in Definition 6.5, and let $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ be the polytope defined by (6.1). If $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is any solution to the* GDAFixedPoint *problem with inputs α, G, L, $\mathcal{C}_{f_{\mathcal{C}_l}}$, $\boldsymbol{A}$, and $\boldsymbol{b}$, defined in (+) then the following statements hold, where recall that $\Delta = \delta/12$. For $i \in \{1, 2\}$:*

- *If $x_i^\star \in (\alpha, 1-\alpha)$ and $x_i^\star \in (y_i^\star - \Delta + \alpha, y_i^\star + \Delta - \alpha)$ then $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i}\right| \leq \alpha$.*
- *If $x_i^\star \leq \alpha$ or $x_i^\star \leq y_i^\star - \Delta + \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} \geq -\alpha$.*
- *If $x_i^\star \geq 1 - \alpha$ or $x_i^\star \geq y_i^\star + \Delta - \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} \leq \alpha$.*

The symmetric statements for $y_i^\star$ hold. For $i \in \{1, 2\}$:

- *If $y_i^\star \in (\alpha, 1-\alpha)$ and $y_i^\star \in (x_i^\star - \Delta + \alpha, x_i^\star + \Delta - \alpha)$ then $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i}\right| \leq \alpha$.*
- *If $y_i^\star \leq \alpha$ or $y_i^\star \leq x_i^\star - \Delta + \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i} \leq \alpha$.*
- *If $y_i^\star \geq 1 - \alpha$ or $y_i^\star \geq x_i^\star + \Delta - \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i} \geq -\alpha$.*

Proof. For this proof it is convenient to define $\hat{\boldsymbol{x}} = \boldsymbol{x}^\star - \nabla_x f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$, $K(\boldsymbol{y}^\star) = \{\boldsymbol{x} \mid (\boldsymbol{x}, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}))\}$, and $\boldsymbol{z} = \Pi_{K(\boldsymbol{y}^\star)}\hat{\boldsymbol{x}}$.

We first consider the first statement, so for the sake of contradiction let's assume that $x_i^\star \in (\alpha, 1-\alpha)$, that $x_i^\star \in (y_i^\star - \Delta + \alpha, y_i^\star + \Delta - \alpha)$, and that $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i}\right| > \alpha$. Due to the definition of $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ in (6.1) the set $K(\boldsymbol{y}^\star)$ is an axes aligned box of $\mathbb{R}^2$ and hence the projection of any vector $\boldsymbol{x}$ onto $K(\boldsymbol{y}^\star)$ can be implemented independently for every coordinate x_i of $\boldsymbol{x}$.

Therefore if it happens that $\hat{x}_i \in (0,1) \cap (y_i^\star - \Delta, y_i^\star + \Delta)$, then it holds that $\hat{x}_i = \boldsymbol{z}_i$. Now from the definition of $\hat{x}_i$ and z_i, and the fact that $K(\boldsymbol{y}^\star)$ is an axes aligned box, we get that $|x_i^\star - z_i| = |x_i^\star - \hat{x}_i| = \left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i}\right| > \alpha$ which contradicts the fact that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is a solution to the problem GDAFixedPoint. On the other hand if $\hat{x}_i \notin (y_i^\star - \Delta, y_i^\star + \Delta) \cap (0,1)$ then z_i has to be on the boundary of $K(\boldsymbol{y}^\star)$ and hence z_i has to be equal to either 0, or 1, or $y_i^\star - \Delta$, or $y_i^\star + \Delta$. In any of these cases since we assumed that $x_i^\star \in (\alpha, 1-\alpha)$ and that $x_i^\star \in (y_i^\star - \Delta + \alpha, y_i^\star + \Delta - \alpha)$ we conclude that $|x_i^\star - z_i| > \alpha$ and hence we get again a contradiction with the fact that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is a solution to the problem GDAFixedPoint. Hence we have that $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i}\right| \leq \alpha$.

For the second case, we assume for the sake of contradiction that $x_i^\star \leq \alpha$ and $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} < -\alpha$. These imply that $\hat{x}_i > x_i^\star + \alpha$ and that $z_i = \min(y_i^\star + \Delta, \hat{x}_i, 1) > \min(\Delta, \hat{x}_i, 1) \geq \min(3\alpha, x_i^\star + \alpha)$. As a result, $|x_i^\star - z_i| = z_i - x_i^\star > \min(3\alpha, \hat{x}_i + \alpha) - x_i^\star$ which is greater than α. The latter is a contradiction with the assumption that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is a solution to the GDAFixedPoint problem. Also if we assume that $x_i^\star \leq y_i^\star - \Delta + \alpha$ using the same reasoning we get that $z_i = \min(\hat{x}_i, y_i^\star + \Delta - \alpha, 1)$. From this we can again prove that $|x_i^\star - z_i| > \alpha$ which contradicts the fact that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is a solution to GDAFixedPoint.

The third case can be proved using the same arguments as the second case. Then using the corresponding arguments we can prove the corresponding statements for the y variables. □

We are now ready to prove that solutions of GDAFixedPoint can only occur in cells that are either panchromatic or violate the boundary conditions of a proper coloring. For convenience in the rest of this section we define $R(\boldsymbol{x})$ to be the cell of the $2^n \times 2^n$ grid that contains $\boldsymbol{x}$.

$$R(\boldsymbol{x}) = \left[\frac{i}{2^n - 1}, \frac{i+1}{2^n - 1}\right] \times \left[\frac{j}{2^n - 1}, \frac{j+1}{2^n - 1}\right], \tag{6.2}$$

for i, j such that $x_1 \in \left[\frac{i}{2^n-1}, \frac{i+1}{2^n-1}\right]$ and $x_2 \in \left[\frac{j}{2^n-1}, \frac{j+1}{2^n-1}\right]$ if there are multiple i, j that satisfy the above condition then we choose $R(\boldsymbol{x})$ to be the cell that corresponds to the i, j such that the pair (i,j) it the lexicographically first such that i, j satisfy the above condition. We also define the corners $R_c(\boldsymbol{x})$ of $R(\boldsymbol{x})$ as

$$R_c(\boldsymbol{x}) = \{(i,j), (i,j+1), (i+1,j), (i+1), (j+1)\} \tag{6.3}$$

where $R(\boldsymbol{x}) = \left[\frac{i}{2^n-1}, \frac{i+1}{2^n-1}\right] \times \left[\frac{j}{2^n-1}, \frac{j+1}{2^n-1}\right]$.

Lemma 6.8. *Let $\mathcal{C}_l$ be an input to the* 2D-BiSperner *problem, let $f_{\mathcal{C}_l}$ be the corresponding G-Lipschitz and L-smooth function defined in Definition 6.5, and let $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ be the polytope defined by (6.1). If $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is any solution to the* GDAFixedPoint *problem with inputs α, G, L, $\mathcal{C}_{f_{\mathcal{C}_l}}$, $\boldsymbol{A}$, and $\boldsymbol{b}$ defined in (+) then none of the following statements hold for the cell $R(\boldsymbol{x}^\star)$.*

1. *$x_1^\star \geq 1/(2^n - 1)$ and, for all $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$, it holds that $\mathcal{C}_l^1(\boldsymbol{v}) = -1$.*
2. *$x_1^\star \leq (2^n - 2)/(2^n - 1)$ and, for all $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$, it holds that $\mathcal{C}_l^1(\boldsymbol{v}) = +1$.*
3. *$x_2^\star \geq 1/(2^n - 1)$ and, for all $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$, it holds that $\mathcal{C}_l^2(\boldsymbol{v}) = -1$.*
4. *$x_2^\star \leq (2^n - 2)/(2^n - 1)$ and, for all $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$, it holds that $\mathcal{C}_l^2(\boldsymbol{v}) = +1$.*

Proof. We prove that there is no solution $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ of GDAFixedPoint that satisfies the statement 1. and the fact that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ cannot satisfy the other statements follows similarly. It is convenient for us to define $\hat{\boldsymbol{x}} = \boldsymbol{x}^\star - \nabla_x f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$, $K(\boldsymbol{y}^\star) = \{\boldsymbol{x} \mid (\boldsymbol{x}, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}))\}$, $\boldsymbol{z} = \Pi_{K(\boldsymbol{y}^\star)}\hat{\boldsymbol{x}}$, and $\hat{\boldsymbol{y}} = \boldsymbol{y}^\star + \nabla_y f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$, $K(\boldsymbol{x}^\star) = \{\boldsymbol{y} \mid (\boldsymbol{x}^\star, \boldsymbol{y}) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}))\}$, $\boldsymbol{w} = \Pi_{K(\boldsymbol{x}^\star)}\hat{\boldsymbol{y}}$.

For the sake of contradiction we assume that there exists a solution of $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ such that $x_1^\star \geq 1/(2^n - 1)$ and for all $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$ it holds that $\mathcal{C}_l^1(\boldsymbol{v}) = -1$. Using the fact that the first color of all the corners of $R(\boldsymbol{x}^\star)$ is 1^-, we will prove that (1) $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_1} \geq 1/2$, and (2) $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_1} = -1$.

Let $R(\boldsymbol{x}^\star) = \left[\frac{i}{2^n-1}, \frac{i+1}{2^n-1}\right] \times \left[\frac{j}{2^n-1}, \frac{j+1}{2^n-1}\right]$, then since all the corners $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$ have $\mathcal{C}_l^1(\boldsymbol{v}) = -1$, from the Definition 6.5 we have that

$$
\begin{aligned}
f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star) = (x_1^\star - y_1^\star) &- (x_2^\star - y_2^\star) \cdot S_1\left(\frac{x_1^C - x_1^\star}{\delta}\right) \cdot S_1\left(\frac{x_2^C - x_2^\star}{\delta}\right) \cdot \mathcal{C}_l^2(i, j) \\
&- (x_2^\star - y_2^\star) \cdot S_1\left(\frac{x_1^D - x_1^\star}{\delta}\right) \cdot S_1\left(\frac{x_2^\star - x_2^D}{\delta}\right) \cdot \mathcal{C}_l^2(i, j+1) \\
&- (x_2^\star - y_2^\star) \cdot S_1\left(\frac{x_1^\star - x_1^A}{\delta}\right) \cdot S_1\left(\frac{x_2^\star - x_2^A}{\delta}\right) \cdot \mathcal{C}_l^2(i+1, j+1) \\
&- (x_2^\star - y_2^\star) \cdot S_1\left(\frac{x_1^\star - x_1^B}{\delta}\right) \cdot S_1\left(\frac{x_2^B - x_2^\star}{\delta}\right) \cdot \mathcal{C}_l^2(i+1, j)
\end{aligned}
$$

where $(x_1^A, x_2^A) = (i/(2^n-1), j/(2^n-1))$, $(x_1^B, x_2^B) = (i/(2^n-1), (j+1)/(2^n-1))$, $(x_1^C, x_2^C) = ((i+1)/(2^n-1), (j+1)/(2^n-1))$, and $(x_1^D, x_2^D) = ((i+1)/(2^n-1), j/(2^n-1))$. If we differentiate this with respect to y_1 we immediately get that $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_1} = -1$. On the other hand if we differentiate with respect to x_1 we get

$$
\begin{aligned}
\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_1} = 1 &+ (x_2^\star - y_2^\star) \cdot \frac{1}{\delta} \cdot S_1'\left(1 - \frac{x_1^\star - x_1^A}{\delta}\right) \cdot S_1\left(1 - \frac{x_2^\star - x_2^A}{\delta}\right) \cdot \mathcal{C}_l^2(i, j) \\
&+ (x_2^\star - y_2^\star) \cdot \frac{1}{\delta} \cdot S_1'\left(1 - \frac{x_1^\star - x_1^A}{\delta}\right) \cdot S_1\left(\frac{x_2^\star - x_2^A}{\delta}\right) \cdot \mathcal{C}_l^2(i, j+1) \\
&- (x_2^\star - y_2^\star) \cdot \frac{1}{\delta} \cdot S_1'\left(\frac{x_1^\star - x_1^A}{\delta}\right) \cdot S_1\left(\frac{x_2^\star - x_2^A}{\delta}\right) \cdot \mathcal{C}_l^2(i+1, j+1) \\
&- (x_2^\star - y_2^\star) \cdot \frac{1}{\delta} \cdot S_1'\left(\frac{x_1^\star - x_1^A}{\delta}\right) \cdot S_1\left(1 - \frac{x_2^\star - x_2^A}{\delta}\right) \cdot \mathcal{C}_l^2(i+1, j) \\
\geq 1 &- 4\,|x_2^\star - y_2^\star| \cdot \frac{3}{2\delta} \\
\geq 1 &- 6 \cdot \frac{\Delta}{\delta} \geq 1/2
\end{aligned}
\tag{6.4}
$$

where the last inequality follows from the fact that $|S_1'(\cdot)| \leq 3/2$ and the fact that, due to the constraints that define the polytope $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$, it holds that $|x_2 - y_2| \leq \Delta$.

Hence we have established that if $x_1^\star \geq 1/(2^n - 1)$ and for all $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$ it holds that $\mathcal{C}_l^1(\boldsymbol{v}) = -1$ then it holds that that (1) $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_1} \geq 1/2$, and (2) $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_1} = -1$. Now it is easy to see that the only way to satisfy both $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_1} \geq 1/2$ and $|z_1 - x_1^\star| \leq \alpha$ is that either $x_1^\star \leq \alpha$ or

$x_1^\star \leq y_1^\star - \Delta + \alpha$. The first case is excluded by the assumption in the first statement of our lemma and our choice of $\alpha = \Delta/3 = 1/(36 \cdot (2^n - 1))$ thus it holds that $x_1^\star \leq y_1^\star - \Delta + \alpha$. But then we can use the case 3 for the y variables of Lemma 6.7 and we get that $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_1} \geq -\alpha$, which cannot be true since we proved that $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_1} = -1$. Therefore we have a contradiction and the first statement of the lemma holds. Using the same reasoning we prove the rest of the statements. □

Remark 6.9. The computations presented in (6.4) is the precise point where an attempt to prove the hardness of minimization problems would fail. In particular, if our goal was to construct a hard minimization instance then the function $f_{\mathcal{C}_l}$ would need to have the terms $x_i + y_i$ instead of $x_i - y_i$ so that the fixed points of gradient descent coincide with approximate local minimum of $f_{\mathcal{C}_l}$. In that case we cannot lower bound the gradient of (6.4) below from $1/2$ because the term $|x_2^\star + y_2^\star|$ will be the dominant one and hence the sign of the derivative can change depending on the value $|x_2^\star + y_2^\star|$. For a more intuitive explanation of the reason why we cannot prove hardness of minimization problems we refer to the Introduction, at Section 1.2.

We have now all the ingredients to prove Theorem 6.1.

Proof of Theorem 6.1. Let $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ be a solution to the GDAFixedPoint instance that we construct based on the instance $\mathcal{C}_l$ of 2D-BiSperner. Let also $R(\boldsymbol{x}^\star)$ be the cell that contains $\boldsymbol{x}^\star$. If the corners $R_c(\boldsymbol{x}^\star)$ contain all the colors 1^-, 1^+, 2^-, 2^+ then we have a solution to the 2D-BiSperner instance and the Theorem 6.1 follows. Otherwise there is at least one color missing from $R_c(\boldsymbol{x}^\star)$, let's assume without loss of generality that one of the missing colors is 1^-, hence for every $\boldsymbol{v} \in R_c(\boldsymbol{x}^\star)$ it holds that $\mathcal{C}_l(\boldsymbol{v}) = +1$. Now from Lemma 6.8 the only way for this to happen is that $x_1^\star > (2^n - 2)/(2^n - 1)$ which implies that in $R_c(\boldsymbol{x}^\star)$ there is at least one corner of the form $\boldsymbol{v} = (2^n - 1, j)$. But we have assumed that $\mathcal{C}_l(\boldsymbol{v}) = +1$, hence $\boldsymbol{v}$ is a violation of the proper coloring rules and hence a solution to the 2D-BiSperner instance. We can prove the corresponding statement if any other color from 1^+, 2^-, 2^+ is missing. Finally, we observe that the function that we define has range $[-2, 2]$ and hence the Theorem 6.1 follows. □

7 Hardness of Local Min-Max Equilibrium – High-Dimensions

Although the results of Section 6 are quite indicative about the computational complexity of GDAFixedPoint and LR-LocalMinMax, we have not yet excluded the possibility of the existence of algorithms running in $\text{poly}(d, G, L, 1/\varepsilon)$ time. In this section we present a, significantly more challenging, high dimensional version of the reduction that we presented in Section 6. The advantage of this reduction is that it rules out the existence even of algorithms running in $\text{poly}(d, G, L, 1/\varepsilon)$ steps unless FP = PPAD, for details see Theorem 4.4. An easy consequence of our result is an unconditional lower bound on the *black-box model* that states that the running time of any algorithm for LR-LocalMinMax that has only oracle access to f and ∇f has to be exponential in d, or G, or L, or $1/\varepsilon$, for details we refer to the Theorem 4.5 and Section 9.

The main reduction that we use to prove Theorem 4.4 is from the high dimensional generalization of the problem 2D-BiSperner, which we call HighD-BiSperner, to GDAFixedPoint. Our reduction in this section resembles some of the ideas of the reductions of Section 6 but it has many additional significant technical difficulties. The main difficulty that we face is how to define a function on a d-dimensional simplex that is: (1) both Lipschitz and smooth, (2) interpolated between some fixed functions at the $d+1$ corners of the simplex, and (3) remains Lipschitz and

smooth even if we glue together different simplices. It is well understood from previous works how to construct such a function if we are interested only in achieving the Lipschitz continuity. Surprisingly adding the smoothness requirement makes the problem very different and significantly more difficult. Our proof overcomes this technical difficulty by introducing a novel but very technically involved way to define interpolation within a simplex of some fixed functions on the corners of the simplex. We believe that our novel interpolation technique is of independent interest and we hope that it will be at the heart of other computational hardness results of optimization problems in continuous optimization.

7.1 The High Dimensional Bi-Sperner Problem

We start by presenting the HighD-BiSperner problem. The HighD-BiSperner is a straightforward d-dimensional generalization of the 2D-BiSperner that we defined in the Section 6. Assume that we have a d-dimensional grid $N \times \cdots (d \text{ times}) \cdots \times N$. We assign to every vertex of this grid a sequence of d colors and we say that a coloring is *proper* if the following rules are satisfied.

1. The ith color of every vertex is either the color i^+ or the color i^-.
2. All the vertices whose ith coordinate is 0, i.e. they are at the lower boundary of the ith direction, should have the ith color equal to i^+.
3. All the vertices whose ith coordinate is 1, i.e. they are at the higher boundary of the ith direction, should have the ith color equal to i^-.

Using proof ideas similar to the proof of the original Sperner's Lemma it is not hard to prove via a combinatorial argument that in every proper coloring of a d-dimensional grid, there exists a cubelet of the grid where all the $2 \cdot d$ colors $\{1^-, 1^+, \ldots, d^-, d^+\}$ appear in some of its vertices, we call such a cubelet *panchromatic*. In the HighD-BiSperner problem we are asked to find such a cubelet, or a violation of the rules of proper coloring. As in Section 6.1 we do not present this combinatorial argument in this paper since the totality of the HighD-BiSperner problem will follow from our reduction from HighD-BiSperner to GDAFixedPoint and our proofs in Section 5 that establish the totality of GDAFixedPoint.

As in the case of 2D-BiSperner, in order to formally define the computational problem HighD-BiSperner we need to define the coloring of the d-dimensional grid $N \times \cdots \times N$ in a succinct way. The fundamental difference compared to the definition of 2D-BiSperner is that for the HighD-BiSperner we assume that N is only *polynomially large*. This difference will enable us to exclude algorithms for GDAFixedPoint that run in time $\text{poly}(d, 1/\alpha, G, L)$. The input to HighD-BiSperner is a coloring via a binary circuit $\mathcal{C}_l$ that takes as input the coordinates of a vertex of the grid and outputs the sequence of colors that are used to color this vertex. Each one of d coordinates is given via the binary representation of a number in $[N] - 1$. Setting $N = 2^\ell$, where here ℓ is a logarithmically in d small number, we have that the representation of each coordinate is a member of $\{0,1\}^\ell$. In the rest of the section we abuse the notation and we use a coordinate $i \in \{0,1\}^\ell$ both as a binary string and as a number in $\left[2^\ell\right] - 1$ and which of the two we use it is clear from the context. The output of $\mathcal{C}_l$ should be a sequence of d colors, where the ith member of this sequence is one of the colors $\{i^-, i^+\}$. We represent this sequence as a member of $\{-1, +1\}^d$, where the ith coordinate refers to the choice of i^- or i^+.

In the definition of the computational problem HighD-BiSperner the input is a circuit $\mathcal{C}_l$, as we described above. As we discussed above in the HighD-BiSperner problem we are asking for

a panchromatic cubelet of the grid. One issue with this high-dimensional setting is that in order to check whether a cubelet is panchromatic or not we have to query all the 2^d corners of this cubelet which makes the verification problem inefficient and hence a containment to the PPAD class cannot be proved. For this reason as a solution to the HighD-BiSperner we ask not just for a cubelet but for $2 \cdot d$ vertices $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(d)}$ $\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(d)}$, not necessarily different, such that they all belong to the same cubelet and the ith output of $\mathcal{C}_l$ with input $\boldsymbol{v}_i$ is -1, i.e. corresponds to the color i^-, whereas the ith output of $\mathcal{C}_l$ with input $\boldsymbol{u}_i$ is $+1$, i.e. corresponds to the color i^+. This way we have a certificate of size $2 \cdot d$ that can be checked in polynomial time. Another possible solution of HighD-BiSperner is a vertex whose coloring violates the aforementioned boundary conditions 2. and 3.. of a proper coloring. For notational convenience we refer to the ith coordinate of $\mathcal{C}_l$ by $\mathcal{C}_l^i$. The formal definition of HighD-BiSperner is then the following.

HighD-BiSperner.
Input: A boolean circuit $\mathcal{C}_l : \underbrace{\{0,1\}^\ell \times \cdots \times \{0,1\}^\ell}_{d \text{ times}} \to \{-1,1\}^d$

Output: One of the following:

1. Two sequences of d vertices $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(d)}$ an $\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(d)}$ with $\boldsymbol{v}^{(i)}, \boldsymbol{u}^{(i)} \in \left(\{0,1\}^\ell\right)^d$ such that $\mathcal{C}_l^i(\boldsymbol{v}^{(i)}) = -1$ and $\mathcal{C}_l^i(\boldsymbol{u}^{(i)}) = +1$.
2. A vertex $\boldsymbol{v} \in \left(\{0,1\}^\ell\right)^d$ with $v_i = \mathbf{0}$ such that $\mathcal{C}_l^i(\boldsymbol{v}) = -1$.
3. A vertex $\boldsymbol{v} \in \left(\{0,1\}^\ell\right)^d$ with $v_i = \mathbf{1}$ such that $\mathcal{C}_l^i(\boldsymbol{v}) = +1$.

Our first step is to establish the PPAD-hardness of HighD-BiSperner in Theorem 7.2. To prove this we use a stronger version of the Brouwer problem that is called γ-SuccinctBrouwer and was first introduced in [Rub16].

γ-SuccinctBrouwer.
Input: A polynomial-time Turing machine $\mathcal{C}_M$ evaluating a $1/\gamma$-Lipschitz continuous vector-valued function $M : [0,1]^d \to [0,1]^d$.
Output: A point $\boldsymbol{x}^\star \in [0,1]^d$ such that $\|M(\boldsymbol{x}^\star) - \boldsymbol{x}^\star\|_2 \leq \gamma$.

Theorem 7.1 ([Rub16]). *γ-SuccinctBrouwer is PPAD-complete for any fixed constant $\gamma > 0$.*

Theorem 7.2. *There is a polynomial time reducton from any instance of the γ-SuccinctBrouwer problem to an instance of HighD-BiSperner with $N = \Theta(d/\gamma^2)$.*

Proof. Consider the function $g(\boldsymbol{x}) = M(\boldsymbol{x}) - \boldsymbol{x}$. Since M is $1/\gamma$-Lipschitz, $g : [0,1]^d \to [-1,1]^d$ is also $(1 + 1/\gamma)$-Lipschitz. Additionally g can be easily computed via a polynomial-time Turing machine $\mathcal{C}_g$ that uses $\mathcal{C}_M$ as a subroutine. We construct the coloring sequences of every vertex of a d-dimensional grid with $N = \Theta(d/\gamma^2)$ points in every direction using g. Let $g_\eta : [0,1]^2 \to [-1,1]^2$ be the function that the Turing Machine $\mathcal{C}_g$ evaluate when the requested accuracy is $\eta > 0$. For each vertex $\boldsymbol{v} = (v_1, \ldots, v_n) \in ([N]-1)^d$ of the d-dimensional grid its coloring sequence $\mathcal{C}_l(\boldsymbol{v}) \in \{-1,1\}^d$ is constructed as follows: For each coordinate $j = 1, \ldots, d$,

$$\mathcal{C}_l^j(\boldsymbol{v}) = \begin{cases} 1 & v_j = 0 \\ -1 & v_j = 2^n - 1 \\ \operatorname{sign}\left(g_j\left(\frac{v_1}{N-1}, \ldots, \frac{v_n}{N-1}\right)\right) & \text{otherwise} \end{cases},$$

where sign : $[-1,1] \mapsto \{-1,1\}$ is the sign function and $g_{\eta,j}(\cdot)$ is the j-th coordinate of g_η. Observe that since $M : [0,1]^d \to [0,1]^d$, for any vertex $\boldsymbol{v}$ with $v_j = 0$ it holds that $\mathcal{C}_l^j(\boldsymbol{v}) = +1$ and respectively for any vertex $\boldsymbol{v}$ with $v_j = N-1$ it holds that $\mathcal{C}_l^j(\boldsymbol{v}) = -1$ due to the fact that the value of M is always in $[0,1]^d$ and hence there are no vertices in the grid satisfying the possible outputs 2. or 3. of the HighD-BiSperner problem. Thus the only possible solution of the above HighD-BiSperner instance is a sequence of $2d$ vertices $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(d)}, \boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(d)}$ on the same cubelet that certify that the corresponding cubelet is panchromatic, as per possible output 1. of the HighD-BiSperner problem. We next prove that any vertex $\boldsymbol{v}$ of that cubelet it holds that

$$\left| g_j\left(\frac{\boldsymbol{v}}{N-1}\right)\right| \leq \frac{2\sqrt{d}}{\gamma N} \quad \text{for all coordinates } j = 1, \ldots, d.$$

Let $\boldsymbol{v}$ be any vertex on the same cubelet with the output vertices $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(d)}, \boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(d)}$. From the guarantees of colors of the sequences $\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(d)}, \boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(d)}$ we have that either $\mathcal{C}_l^j(\boldsymbol{v}) \cdot \mathcal{C}_l^j(\boldsymbol{v}^{(j)}) = -1$ or $\mathcal{C}_l^j(\boldsymbol{v}) \cdot \mathcal{C}_l^j(\boldsymbol{u}^{(j)}) = -1$, let $\overline{\boldsymbol{v}}^{(j)}$ be the vertex $\boldsymbol{v}^{(j)}$ or $\boldsymbol{u}^{(j)}$ depending on which one the jth color has product equal to -1 with $\mathcal{C}_l^j(\boldsymbol{v})$. Now let $\eta = \frac{2\sqrt{d}}{\gamma N}$ if $g_j\left(\frac{\boldsymbol{v}}{N-1}\right) \in [-\eta, \eta]$ then the wanted inequality follows. On the other hand if $g_j\left(\frac{\boldsymbol{v}}{N-1}\right) \in [-\eta, \eta]$ then using the fact that $\left\|g\left(\frac{\boldsymbol{v}}{N-1}\right) - g_\eta\left(\frac{\boldsymbol{v}}{N-1}\right)\right\|_\infty \leq \eta$ and that from the definition of the colors we have that either $g_{\eta,j}\left(\frac{\boldsymbol{v}}{N-1}\right) \geq 0$, $g_{\eta,j}\left(\frac{\overline{\boldsymbol{v}}^{(j)}}{N-1}\right) < 0$ or $g_{\eta,j}\left(\frac{\boldsymbol{v}}{N-1}\right) < 0$, $g_{\eta,j}\left(\frac{\hat{\boldsymbol{v}}^{(j)}}{N-1}\right) \geq 0$ we conclude that $g_j\left(\frac{\boldsymbol{v}}{N-1}\right) \geq 0$, $g_j\left(\frac{\overline{\boldsymbol{v}}^{(j)}}{N-1}\right) < 0$ or $g_j\left(\frac{\boldsymbol{v}}{N-1}\right) < 0$, $g_j\left(\frac{\hat{\boldsymbol{v}}^{(j)}}{N-1}\right) \geq 0$ and thus,

$$\left| g_j\left(\frac{\boldsymbol{v}}{N-1}\right)\right| \leq \left| g_j\left(\frac{\boldsymbol{v}}{N-1}\right) - g_j\left(\frac{\overline{\boldsymbol{v}}^{(j)}}{N-1}\right)\right| \leq \left(1 + \frac{1}{\gamma}\right) \cdot \left\|\frac{\boldsymbol{v}}{N-1} - \frac{\overline{\boldsymbol{v}}^{(j)}}{N-1}\right\|_2 \leq \frac{2\sqrt{d}}{\gamma N}$$

where in the second inequality we have used the $(1 + 1/\gamma)$-Lipschitzness of g. As a result, the point $\hat{\boldsymbol{v}} = \boldsymbol{v}/(N-1) \in [0,1]^d$ satisfies $\|M(\hat{\boldsymbol{v}}) - \hat{\boldsymbol{v}}\|_2 \leq 2d/(\gamma N)$ and thus for if we pick $N = \Theta(d/\gamma^2)$ then any vertex $\boldsymbol{v}$ of the panchromatic cell is a solution for γ-SuccinctBrouwer. □

Now that we have established the PPAD-hardness of HighD-BiSperner we are ready to present our main result of this section which is a reduction from the problem HighD-BiSperner to the problem GDAFixedPoint with the additional constraints that the scalars α, G, L in the input satisfy $1/\alpha = \text{poly}(d)$, $G = \text{poly}(d)$, and $L = \text{poly}(d)$.

7.2 From High Dimensional Bi-Sperner to Fixed Points of Gradient Descent/Ascent

Given the binary circuit $\mathcal{C}_l : ([N]-1)^d \to \{-1,+1\}^d$ that is an instance of HighD-BiSperner, we construct a G-Lipschitz and L-smooth function $f_{\mathcal{C}_l} : [0,1]^d \times [0,1]^d \to \mathbb{R}$. To do so, we divide the $[0,1]^d$ hypercube into cubelets of length $\delta = 1/(N-1)$. The corners of such cubelets have coordinates that are integer multiples of $\delta = 1/(N-1)$ and we call them *vertices*. Each vertex can be represented by the vector $\boldsymbol{v} = (v_1, \ldots, v_d) \in ([N]-1)^d$ and admits a coloring sequence defined by the boolean circuit $\mathcal{C}_l : ([N]-1)^d \to \{-1,+1\}^d$. For every $\boldsymbol{x} \in [0,1]^d$, we use $R(\boldsymbol{x})$ to denote the cubelet that contains $\boldsymbol{x}$, formally

$$R(\boldsymbol{x}) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$$

where $\boldsymbol{c} \in ([N-1]-1)^d$ such that $\boldsymbol{x} \in \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$ and if there are multiple corners $\boldsymbol{c}$ that satisfy this condition then we choose $R(\boldsymbol{x})$ to be the cell that corresponds to the $\boldsymbol{c}$

that is lexicographically first among those that satisfy the condition. We also define $R_c(\boldsymbol{x})$ to be the set of vertices that are corners of the cublet $R(\boldsymbol{x})$, namely

$$R_c(\boldsymbol{x}) = \{c_1, c_1+1\} \times \cdots \times \{c_d, c_d+1\}$$

where $\boldsymbol{c} \in ([N-1]-1)^d$ such that $R(\boldsymbol{x}) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$ Every $\boldsymbol{y}$ that belongs to the cubelet $R(\boldsymbol{x})$ can be written as a convex combination of the vectors $\boldsymbol{v}/(N-1)$ where $\boldsymbol{v} \in R_c(\boldsymbol{x})$. The value of the function $f_{\mathcal{C}_l}(\boldsymbol{x}, \boldsymbol{y})$ that we construct in this section is determined by the coloring sequences $\mathcal{C}_l(\boldsymbol{v})$ of the vertices $\boldsymbol{v} \in R_c(\boldsymbol{x})$. One of the main challenges that we face though is that the size of $R_c(\boldsymbol{x})$ is 2^d and hence if we want to be able to compute the value of $f_{\mathcal{C}_l}(\boldsymbol{x}, \boldsymbol{y})$ efficiently then we have to find a consistent rule to pick a subset of the vertices of $R_c(\boldsymbol{x})$ whose coloring sequence we need to define the function value $f_{\mathcal{C}_l}(\boldsymbol{x}, \boldsymbol{y})$. Although there are traditional ways to overcome this difficulty using the *canonical simplicization* of the cubelet $R(\boldsymbol{x})$, these technique leads only to functions that are continuous and Lipschitz but they are not enough to guarantee continuity of the gradient and hence the resulting functions are not smooth.

7.2.1 Smooth and Efficient Interpolation Coefficients

The problem of finding a computationally efficient way to define a continuous function as an interpolation of some fixed function in the corners of a cubelet so that the resulting function is both Lischitz and smooth is surprisingly difficult to solve. For this reason we introduce in this section the *smooth and efficient interpolation coefficients (SEIC)* that as we will see in Section 7.2.2, is the main technical tool to implement such an interpolation. Our novel interpolation coefficients are of independent interest and we believe that they will serve as a main technical tool for proving other hardness results in continuous optimization in the future.

In this section we only give a high level description of the smooth and efficient interpolation coefficients via their properties that we use in Section 7.2.2 to define the function $f_{\mathcal{C}_l}$. The actual construction of the coefficients is very challenging and technical and hence we postpone a detail exposition for Section 8.

Definition 7.3 (Smooth and Efficient Interpolation Coefficients)**.** For every $N \in \mathbb{N}$ we define the set of *smooth and efficient interpolation coefficients (SEIC)* as the family of functions, called *coefficients*, $\mathcal{I}_{d,N} = \left\{\mathsf{P}_{\boldsymbol{v}} : [0,1]^d \to \mathbb{R} \mid \boldsymbol{v} \in ([N]-1)^d\right\}$ with the following properties.

(A) For all vertices $\boldsymbol{v} \in ([N]-1)^d$, the coefficient $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})$ is a twice-differentiable function and satisfies

- $\left|\frac{\partial \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})}{\partial x_i}\right| \leq \Theta(d^{12}/\delta)$.
- $\left|\frac{\partial^2 \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})}{\partial x_i\, \partial x_\ell}\right| \leq \Theta(d^{24}/\delta^2)$.

(B) For all $\boldsymbol{v} \in ([N]-1)^d$, it holds that $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) \geq 0$ and $\sum_{\boldsymbol{v} \in ([N]-1)^d} \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = \sum_{\boldsymbol{v} \in R_c(\boldsymbol{x})} \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 1$.

(C) For all $\boldsymbol{x} \in [0,1]^d$, it holds that all but $d+1$ of the coefficients $\mathsf{P}_{\boldsymbol{v}} \in \mathcal{I}_{d,N}$ satisfy $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 0$, $\nabla \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 0$ and $\nabla^2 \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 0$. We denote this set of $d+1$ vertices by $R_+(\boldsymbol{x})$. Furthermore, it holds that $R_+(\boldsymbol{x}) \subseteq R_c(\boldsymbol{x})$ and given $\boldsymbol{x}$ we can compute the set $R_+(\boldsymbol{x})$ it time poly(d).

(D) For all $\boldsymbol{x} \in [0,1]^d$, if $x_i \leq 1/(N-1)$ for some $i \in [d]$ then there exists $\boldsymbol{v} \in R_+(\boldsymbol{x})$ such that $v_i = 0$. Respectively, if $x_i \geq 1 - 1/(N-1)$ then there exists $\boldsymbol{v} \in R_+(\boldsymbol{x})$ such that $v_i = 1$.

An intuitive explanation of the properties of the SEIC coefficients is the following

(A) – The coefficients P_v are both Lipschitz and smooth with Lipschitzness and smoothness parameters that depends polynomially in d and $N = 1/\delta + 1$.

(B) – The coefficients $\mathsf{P}_v(\boldsymbol{x})$ define a convex combination of the vertices $R_c(\boldsymbol{x})$.

(C) – For every $\boldsymbol{x} \in [0,1]^d$, out of the N^d coefficients P_v only $d+1$ have non-zero value, or non-zero gradient or non-zero Hessian when evaluated at the point $\boldsymbol{x}$. Moreover, given $\boldsymbol{x} \in [0,1]^d$ we can identify these $d+1$ coefficients efficiently.

(D) – For every $\boldsymbol{x} \in [0,1]^d$ that is in a cubelet that touches the boundary there is at least one of the vertices in $R_+(\boldsymbol{x})$ that is on the boundary of the continuous hypercube $[0,1]^d$.

In Section 10 in the proof of Theorem 10.4 we present a simple application of the existence of the SEIC coefficients for proving very simple black box oracle lower bounds for the global minimization problem.

Based on the existence of these coefficients we are now ready to define the function $f_{\mathcal{C}_l}$ which is the main construction of our reduction.

7.2.2 Definition of a Lipschitz and Smooth Function Based on a BiSperner Instance

In this section our goal is to formally define the function $f_{\mathcal{C}_l}$ and prove its Lipschitzness and smoothness properties in Lemma 7.5.

Definition 7.4 (Continuous and Smooth Function from Colorings of Bi-Sperner)**.** Given a binary circuit $\mathcal{C}_l : ([N]-1)^d \to \{-1,1\}^d$, we define the function $f_{\mathcal{C}_l} : [0,1]^d \times [0,1]^d \to \mathbb{R}$ as follows

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

where $\alpha_j(\boldsymbol{x}) = -\sum_{\boldsymbol{v}\in([N]-1)^d} \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) \cdot \mathcal{C}_l^j(\boldsymbol{v})$, and $\mathsf{P}_{\boldsymbol{v}}$ are the coefficients defined in Definition 7.3.

We first prove that the function $f_{\mathcal{C}_l}$ constructed in Definition 7.4 is G-Lipschitz and L-smooth for some appropriately selected parameters G, L that are polynomial in the dimension d and in the discretization parameter N. We use this property to establish that $f_{\mathcal{C}_l}$ is a valid input to the promise problem GDAFixedPoint.

Lemma 7.5. *The function $f_{\mathcal{C}_l}$ of Definition 7.4 is $O(d^{15}/\delta)$-Lipschitz and $O(d^{27}/\delta^2)$-smooth.*

Proof. If we take the derivative with respect to x_i and y_i and using property (B) of the coefficients $\mathsf{P}_{\boldsymbol{v}}$ we get the following relations,

$$\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial x_i} = \sum_{j=1}^{d} (x_j - y_j) \cdot \frac{\partial \alpha_j(\boldsymbol{x})}{\partial x_i} + \alpha_i(\boldsymbol{x}) \quad \text{and} \quad \frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial y_i} = -\alpha_i(\boldsymbol{x})$$

where

$$\alpha_i(\boldsymbol{x}) = -\sum_{\boldsymbol{v}\in([N]-1)^d} \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) \quad \text{and} \quad \frac{\partial \alpha_j(\boldsymbol{x})}{\partial x_i} = -\sum_{\boldsymbol{v}\in([N]-1)^d} \frac{\partial \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})}{\partial x_i} \cdot \mathcal{C}_l^j(\boldsymbol{v}).$$

Now by the property (C) of Definition 7.3 there are most $d+1$ vertices $\boldsymbol{v}$ of $R_c(\boldsymbol{x})$ with the property $\nabla \mathsf{P}_v(\boldsymbol{x}) \neq 0$. Then if we also use property (A) we get $\left|\frac{\partial \alpha_j(\boldsymbol{x})}{\partial x_i}\right| \leq \Theta(d^{13}/\delta)$ and using the property (B) we get $|\alpha_i(\boldsymbol{x})| \leq 1$. Thus $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial x_i}\right| \leq \Theta(d^{14}/\delta)$ and $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial y_i}\right| \leq \Theta(d)$. Therefore we can conclude that $\left\|\nabla f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})\right\|_2 \leq \Theta(d^{15}/\delta)$ and hence this proves that the function $f_{\mathcal{C}_l}$ is Lipschitz continuous with Lipschitz constant $\Theta(d^{15}/\delta)$.

To prove the smoothness of $f_{\mathcal{C}_l}$, we use the property (B) of the Definition 7.3 and we have

$$\frac{\partial^2 f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial x_i\, \partial x_\ell} = \sum_{j=1}^{d}(x_j - y_j)\cdot \frac{\partial^2 \alpha_j(\boldsymbol{x})}{\partial x_i\, \partial x_\ell} + \frac{\partial \alpha_\ell(\boldsymbol{x})}{\partial x_i} + \qquad \frac{\partial \alpha_i(\boldsymbol{x})}{\partial x_\ell},$$

$$\frac{\partial^2 f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial x_i\, \partial y_\ell} = -\frac{\partial \alpha_\ell(\boldsymbol{x})}{\partial x_i}, \quad \text{and} \quad \frac{\partial^2 f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})}{\partial y_i\, \partial y_\ell} = 0$$

where

$$\frac{\partial^2 \alpha_j(\boldsymbol{x})}{\partial x_i\, \partial x_\ell} = -\sum_{v\in([N]-1)^d} \frac{\partial^2 \mathsf{P}_v(\boldsymbol{x})}{\partial x_i\, \partial x_\ell}\cdot \mathcal{C}_l^j(\boldsymbol{v})$$

Again using the property (C) of Definition 7.3 we get that there are most $d+1$ vertices $\boldsymbol{v}$ of $R_c(\boldsymbol{x})$ such that $\nabla^2 \mathsf{P}_v(\boldsymbol{x}) \neq 0$. This together with the property (A) of Definition 7.3 leads to the fact that $\left|\frac{\partial^2 \alpha_j(\boldsymbol{x})}{\partial x_i\, \partial x_\ell}\right| \leq \Theta(d^{25}/\delta^2)$. Using the later together with the bounds that we obtained for $\left|\frac{\partial \alpha_j(\boldsymbol{x})}{\partial x_i}\right|$ in the beginning of the proof we get that $\left\|\nabla^2 f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})\right\|_F \leq \Theta(d^{27}/\delta^2)$, where with $\|\cdot\|_F$ we denote the Frobenious norm. Since the bound on the Frobenious norm is a bound to the spectral norm too, we get that the function $f_{\mathcal{C}_l}$ is $\Theta(d^{27}/\delta^2)$-smooth. □

7.2.3 Description and Correctness of the Reduction – Proof of Theorem 4.4

We start with a description of the reduction from HighD-BiSperner to GDAFixedPoint. Suppose we have an instance of HighD-BiSperner given by boolean circuit $\mathcal{C}_l : ([N]-1)^d \to \{-1,1\}^d$, we construct an instance of GDAFixedPoint according to the following set of rules.

($\star$) Construction of Instance for Fixed Points of Gradient Descent/Ascent.

- The payoff function is the real-valued function $f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})$ from the Definition 7.4.
- The domain is the polytope $\mathcal{P}(\boldsymbol{A},\boldsymbol{b})$ that we described in Section 3. The matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ are computed so that the following inequalities hold

$$x_i - y_i \leq \Delta, \;\; y_i - x_i \leq \Delta \quad \text{for all} \;\; i \in [d] \tag{7.1}$$

  where $\Delta = t\cdot\delta/d^{14}$, with $t \in \mathbb{R}_+$ be a constant such that $\left|\frac{\partial \mathsf{P}_v(\boldsymbol{x})}{\partial x_i}\right| \cdot \frac{\delta}{d^{12}} t \leq \frac{1}{2}$, for all $\boldsymbol{v} \in ([N]-1)^d$ and $\boldsymbol{x} \in [0,1]^d$. The fact that such a constant t exists follows from the property (A) of the smooth and efficient coefficients.
- The parameter α is set to be equal to $\Delta/3$.
- The parameters G and L are set to be equal to the upper bounds on the Lipschitzness and the smoothness of $f_{\mathcal{C}_l}$ respectively that we derived in Lemma 7.5. Namely we have that $G = O(d^{15}/\delta)$ and $L = O(d^{27}/\delta^2)$.

The first thing to observe is that the afore-described reduction is polynomial-time. For this observe that all of α, G, L, $\boldsymbol{A}$, and $\boldsymbol{b}$ have representation that is polynomial in d even if we use unary instead of binary representation. So the only thing that remains is the existence of a Turing machine $\mathcal{C}_{f_{\mathcal{C}_l}}$ that computes the function and the gradient value of $f_{\mathcal{C}_l}$ in time polynomial to the size of $\mathcal{C}_l$ and the requested accuracy. To prove this we need a detailed description of the SEIC coefficients and for this reason we postpone the proof of this to the Appendix D. Here we state the formally the result that we prove in the Appendix D which together with the discussion above proves that our reduction is indeed polynomial-time.

Theorem 7.6. *Given a binary circuit $\mathcal{C}_l : ([N]-1)^d \to \{-1,1\}^d$ that is an input to the* HighD-BiSperner *problem. Then, there exists a polynomial-time Turing machine $\mathcal{C}_{f_{\mathcal{C}_l}}$, that can be constructed in polynomial-time from the circuit $\mathcal{C}_l$ such that for all vector $\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d$ and accuracy $\varepsilon > 0$, $\mathcal{C}_{f_{\mathcal{C}_l}}$ computes both $z \in \mathbb{R}$ and $\boldsymbol{w} \in \mathbb{R}^d$ such that*

$$\left|z - f_{\mathcal{C}_l}(\boldsymbol{x}, \boldsymbol{y})\right| \leq \varepsilon, \quad \left\|\boldsymbol{w} - \nabla f_{\mathcal{C}_l}(\boldsymbol{x}, \boldsymbol{y})\right\|_2 \leq \varepsilon.$$

Moreover the running time of $\mathcal{C}_{f_{\mathcal{C}_l}}$ is polynomial in the binary representation of $\boldsymbol{x}$, $\boldsymbol{y}$, and $\log(1/\varepsilon)$.

We also observe that according to Lemma 7.5, the function $f_{\mathcal{C}_l}$ is both G-Lipschitz and L-smooth and hence the output of our reduction is a valid input for the constructed instance of the promise problem GDAFixedPoint. The next step is to prove that the vector $\boldsymbol{x}^\star$ of every solution $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ of GDAFixedPoint with input as we described above, lies in a cubelet that is either panchromatic according to $\mathcal{C}_l$ or is a violation of the rules for proper coloring of the HighD-BiSperner problem.

Lemma 7.7. *Let $\mathcal{C}_l$ be an input to the* HighD-BiSperner *problem, let $f_{\mathcal{C}_l}$ be the corresponding G-Lipschitz and L-smooth function defined in Definition 7.4, and let $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ be the polytope defined by (7.1). If $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is any solution to the* GDAFixedPoint *problem with input α, G, L, $\mathcal{C}_{f_{\mathcal{C}_l}}$, $\boldsymbol{A}$, and $\boldsymbol{b}$, defined in $(\star)$ then the following statements hold, where we remind that $\Delta = t \cdot \delta / d^{14}$.*

- *If $x_i^\star \in (\alpha, 1-\alpha)$ and $x_i^\star \in (y_i^\star - \Delta + \alpha, y_i^\star + \Delta - \alpha)$ then $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i}\right| \leq \alpha$.*
- *If $x_i^\star \leq \alpha$ or $x_i^\star \leq y_i^\star - \Delta + \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} \geq -\alpha$.*
- *If $x_i^\star \geq 1 - \alpha$ or $x_i^\star \geq y_i^\star + \Delta - \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} \leq \alpha$.*

The symmetric statements for $y_i^\star$ hold.

- *If $y_i^\star \in (\alpha, 1-\alpha)$ and $y_i^\star \in (x_i^\star - \Delta + \alpha, x_i^\star + \Delta - \alpha)$ then $\left|\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i}\right| \leq \alpha$.*
- *If $y_i^\star \leq \alpha$ or $y_i^\star \leq x_i^\star - \Delta + \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i} \leq \alpha$.*
- *If $y_i^\star \geq 1 - \alpha$ or $y_i^\star \geq x_i^\star + \Delta - \alpha$ then $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i} \geq -\alpha$.*

Proof. The proof of this lemma is identical to the proof of Lemma 6.7 and for this reason we skip the details of the proof here. □

Lemma 7.8. *Let $\mathcal{C}_l$ be an input to the* HighD-BiSperner *problem, let $f_{\mathcal{C}_l}$ be the corresponding G-Lipschitz and L-smooth function defined in Definition 7.4, and let $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ be the polytope defined by (7.1). If $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is any solution to the* GDAFixedPoint *problem with input α, G, L, $\mathcal{C}_{f_{\mathcal{C}_l}}$, $\boldsymbol{A}$, and $\boldsymbol{b}$, defined in $(\star)$, then none of the following statements hold for the cubelet $R(\boldsymbol{x}^\star)$.*

1. *$x_i^\star \geq 1/(N-1)$ and for any $\boldsymbol{v} \in R_+(\boldsymbol{x}^\star)$, it holds that $\mathcal{C}_l^i(\boldsymbol{v}) = -1$.*
2. *$x_i^\star \leq 1 - 1/(N-1)$ and for any $\boldsymbol{v} \in R_+(\boldsymbol{x}^\star)$, it holds that $\mathcal{C}_l^1(\boldsymbol{v}) = +1$.*

Proof. We prove that there is no solution $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ of GDAFixedPoint that satisfies the statement 1. and the fact that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ cannot satisfy the statement 2. follows similarly. It is convenient for us to define $\hat{\boldsymbol{x}} = \boldsymbol{x}^\star - \nabla_x f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$, $K(\boldsymbol{y}^\star) = \{\boldsymbol{x} \mid (\boldsymbol{x}, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}))\}$, $\boldsymbol{z} = \Pi_{K(\boldsymbol{y}^\star)}\hat{\boldsymbol{x}}$, and $\hat{\boldsymbol{y}} = \boldsymbol{y}^\star - \nabla_y f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$, $K(\boldsymbol{x}^\star) = \{\boldsymbol{y} \mid (\boldsymbol{x}^\star, \boldsymbol{y}) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}))\}$, $\boldsymbol{w} = \Pi_{K(\boldsymbol{x}^\star)}\hat{\boldsymbol{y}}$.

For the sake of contradiction we assume that there exists a solution of $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ such that $x_1^\star \geq 1/(N-1)$ and for any $\boldsymbol{v} \in R_+(\boldsymbol{x}^\star)$ it holds that $\mathcal{C}_l^i(\boldsymbol{v}) = -1$. Using this fact, we will prove that (1) $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} \geq 1/2$, and (2) $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i} = -1$.

Let $R(\boldsymbol{x}^\star) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$, then since all the corners $\boldsymbol{v} \in R_+(\boldsymbol{x}^\star)$ have $\mathcal{C}_l^i(\boldsymbol{v}) = -1$, from the Definition 7.4 we have that

$$f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star) = (x_i^\star - y_i^\star) + \sum_{j=1, j\neq i}^{d} (x_j^\star - y_j^\star) \cdot \alpha_j(\boldsymbol{x})$$

If we differentiate this with respect to y_i we immediately get that $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i} = -1$. On the other hand if we differentiate with respect to x_i we get

$$\begin{aligned}
\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} &= 1 + \sum_{j=1, j\neq i}^{d} (x_j - y_j) \cdot \frac{\partial \alpha_j(\boldsymbol{x})}{\partial x_i} \\
&\geq 1 - \sum_{j\neq i} |x_j - y_j| \cdot \left|\frac{\partial \alpha_j(\boldsymbol{x})}{\partial x_i}\right| \\
&\geq 1 - \Delta \cdot d \cdot \Theta\left(\frac{d^{13}}{\delta}\right) \\
&\geq 1/2
\end{aligned}$$

where the above follows from the following facts: (1) that $\left|\frac{\partial \alpha_j(\boldsymbol{x})}{\partial x_l}\right| \leq \Theta(d^{13}/\delta)$, which is proved in the proof of Lemma 7.5, (2) $|x_j - y_j| \leq \Delta$, and (3) the definition of Δ. Now it is easy to see that the only way to satisfy both $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial x_i} \geq 1/2$ and $|z_i - x_i^\star| \leq \alpha$ is that either $x_i^\star \leq \alpha$ or $x_i^\star \leq y_i^\star - \Delta + \alpha$. The first case is excluded by the assumption of the first statement of our lemma and our choice of $\alpha = \Delta/3 < 1/(N-1)$, thus it holds that $x_i^\star \leq y_i^\star - \Delta + \alpha$. But then we can use the case 3. for the y variables of Lemma 6.7 and we get that $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_1} \geq -\alpha$, which cannot be true since we proved that $\frac{\partial f_{\mathcal{C}_l}(\boldsymbol{x}^\star, \boldsymbol{y}^\star)}{\partial y_i} = -1$. Therefore we have a contradiction and the first statement of the lemma holds. Using the same reasoning we prove the second statement too. □

We are now ready to complete the proof that the our reduction from HighD-BiSperner to GDAFixedPoint is correct and hence we can prove Theorem 4.4.

Proof of Theorem 4.4. Let $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ be a solution to the GDAFixedPoint problem with input a Turing machine that represents the function $f_{\mathcal{C}_l}$, $\alpha = \Delta/3$, where $\Delta = t \cdot \delta/d^{14}$, $G = \Theta(d^{15}/\delta)$, $L = \Theta(d^{27}/\delta^2)$, and $\boldsymbol{A}$, $\boldsymbol{b}$ as described in $(\star)$.

For each coordinate i, there exist the following three mutually exclusive cases,

- ▷ $\frac{1}{N-1} \leq \boldsymbol{x}_i^\star \leq \mathbf{1} - \frac{1}{N-1}$: Since $|R_+(\boldsymbol{x}^\star)| \geq 1$, it follows directly from Lemma 7.8 that there exists $\boldsymbol{v} \in R_+(\boldsymbol{x}^\star)$ such that $\mathcal{C}_l^i(\boldsymbol{v}) = -1$ and $\boldsymbol{v}' \in R_+(\boldsymbol{x}^\star)$ such that $\mathcal{C}_l^i(\boldsymbol{v}) = +1$.
- ▷ $\mathbf{0} \leq \boldsymbol{x}_i^\star < \frac{1}{N-1}$: Let $\mathcal{C}_l^i(\boldsymbol{v}) = -1$ for all $\boldsymbol{v} \in R_+(\boldsymbol{x}^\star)$. By the property (D) of the SEIC coefficients, we have that there exists $\boldsymbol{v} \in R_+(x^\star)$ with $v_i = 0$. This node is hence a solution of type 2. for the HighD-BiSperner problem.
- ▷ $\mathbf{1} - \frac{1}{N-1} < \boldsymbol{x}_i^\star \leq \mathbf{1}$: Let $\mathcal{C}_l^i(\boldsymbol{v}) = +1$ for all $\boldsymbol{v} \in R_+(\boldsymbol{x}^\star)$. By the property (D) of the SEIC coefficients, we have that there exists $\boldsymbol{v} \in R_+(x^\star)$ with $v_i = 1$. This node is hence a solution of type 3. for the HighD-BiSperner problem.

Since $\mathbb{R}_+(\boldsymbol{x}^\star)$ computable in polynomial time given $\boldsymbol{x}^\star$, we can easily check for every $i \in [d]$ whether any of the above cases hold. If at least for some $i \in [d]$ the 2nd or the 3rd case from above hold, then the corresponding vertex gives a solution to the HighD-BiSperner problem and therefore our reduction is correct. Hence we may assume that for every $i \in [d]$ the 1st of the above cases holds. This implies that the cubelet $R(\boldsymbol{x}^\star)$ is pachromatic and therefore it is a solution to the problem HighD-BiSperner. Finally, we observe that the function that we define has range $[-d, d]$ and hence the Theorem 4.4 follows using Theorem 5.1. □

8 Smooth and Efficient Interpolation Coefficients

In this section we describe the construction of the smooth and efficient interpolation coefficients (SEIC) that we introduced in Section 7.2.1. After the description of the construction we present the statements of the lemmas that prove the properties (A) - (D) of their Definition 7.3 and we refer to the Appendix C. We first remind the definition of the SEIC coefficients.

Definition 7.3 (Smooth and Efficient Interpolation Coefficients)**.** For every $N \in \mathbb{N}$ we define the set of *smooth and efficient interpolation coefficients (SEIC)* as the family of functions, called *coefficients*, $\mathcal{I}_{d,N} = \left\{ \mathsf{P}_{\boldsymbol{v}} : [0,1]^d \to \mathbb{R} \mid \boldsymbol{v} \in ([N]-1)^d \right\}$ with the following properties.

(A) For all vertices $\boldsymbol{v} \in ([N]-1)^d$, the coefficient $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})$ is a twice-differentiable function and satisfies

- ▶ $\left| \frac{\partial \mathsf{P}_v(\boldsymbol{x})}{\partial x_i} \right| \leq \Theta(d^{12}/\delta)$.
- ▶ $\left| \frac{\partial^2 \mathsf{P}_v(\boldsymbol{x})}{\partial x_i \, \partial x_\ell} \right| \leq \Theta(d^{24}/\delta^2)$.

(B) For all $\boldsymbol{v} \in ([N]-1)^d$, it holds that $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) \geq 0$ and $\sum_{\boldsymbol{v} \in ([N]-1)^d} \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = \sum_{\boldsymbol{v} \in R_c(\boldsymbol{x})} \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 1$.

(C) For all $\boldsymbol{x} \in [0,1]^d$, it holds that all but $d+1$ of the coefficients $\mathsf{P}_{\boldsymbol{v}} \in \mathcal{I}_{d,N}$ satisfy $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 0$, $\nabla \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 0$ and $\nabla^2 \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 0$. We denote this set of $d+1$ vertices by $R_+(\boldsymbol{x})$. Furthermore, it holds that $R_+(\boldsymbol{x}) \subseteq R_c(\boldsymbol{x})$ and given $\boldsymbol{x}$ we can compute the set $R_+(\boldsymbol{x})$ it time poly(d).

(D) For all $\boldsymbol{x} \in [0,1]^d$, if $x_i \leq 1/(N-1)$ for some $i \in [d]$ then there exists $\boldsymbol{v} \in R_+(\boldsymbol{x})$ such that $v_i = 0$. Respectively, if $x_i \geq 1 - 1/(N-1)$ then there exists $\boldsymbol{v} \in R_+(\boldsymbol{x})$ such that $v_i = 1$.

Our main goal in this section is to prove the following theorem.

Theorem 8.1. *For every $d \in \mathbb{N}$ and every $N = \text{poly}(d)$ there exist a family of functions $\mathcal{I}_{d,N}$ that satisfies the properties (A) - (D) of Definition 7.3.*

One important component of the construction of the SEIC coefficients is the *smooth-step functions* which we introduce in Section 8.1. These functions also provide a toy example of smooth and efficient interpolation coefficients in 1 dimension. Then in Section 8.2 we present the construction of the SEIC coefficients in multiple dimensions and in Section 8.3 we state the main lemmas that lead to the proof of Theorem 8.1.

8.1 Smooth Step Functions – Toy Single Dimensional Example

Smooth step functions are real-valued function $g : \mathbb{R} \to \mathbb{R}$ of a single real variable with the following properties

Step Value. For every $x \leq 0$ it holds that $g(x) = 0$, for every $x \geq 1$ it holds that $g(x) = 1$ and for every $x \in [0, 1]$ it holds that $S(x) \in [0, 1]$.

Smoothness. For some k it holds that g is k times continuously differentiable and its kth derivative satisfies $g^{(k)}(0) = 0$ and $g^{(k)}(1) = 0$.

The largest number k such that the smoothness property from above holds is characterizes the *order of smoothness* of the smooth step function g.

In Section 6 we have already defined and used the smooth step function of order 1. For the construction of the SEIC coefficients we use the smooth step function of order 2 and the smooth step function of order ∞ defined as follows.

Definition 8.2. We define the smooth step function $S : \mathbb{R} \to \mathbb{R}$ of order 2 as the following function

$$S(x) = \begin{cases} 6x^5 - 15x^4 + 10x^3 & x \in (0,1) \\ 0 & x \leq 0 \\ 1 & x \geq 1 \end{cases}.$$

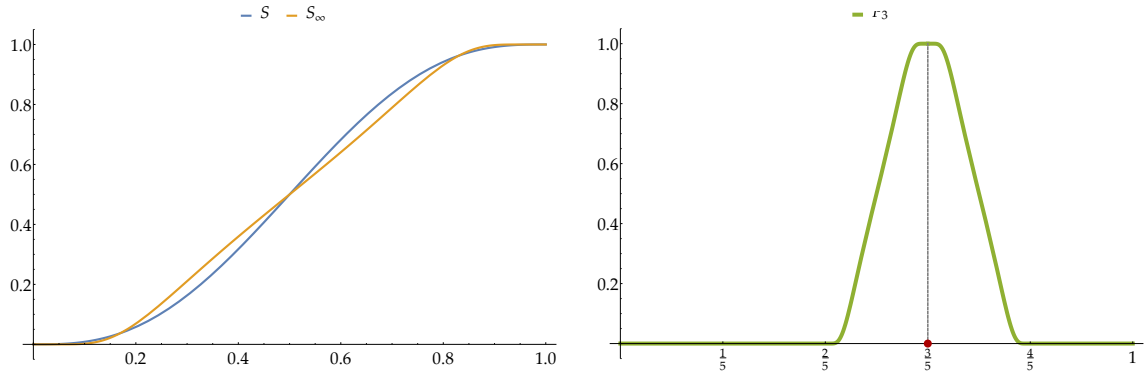
We also define the smooth step function $S_\infty : \mathbb{R} \to \mathbb{R}$ of order ∞ as the following function

$$S_\infty(x) = \begin{cases} \frac{2^{-1/x}}{2^{-1/x} + 2^{-1/(1-x)}} & x \in (0,1) \\ 0 & x \leq 0 \\ 1 & x \geq 1 \end{cases}.$$

We note that we use the notation S instead of S_2 for the smooth step function of order 2 for simplicitly of the exposition of the paper.

We present a plot of these step function in Figure 7, and we summarize some of their properties in Lemma 8.3. A more detailed lemma with additional properties of S_∞ that are useful for the proof of Theorem 8.1 is presented in Lemma C.5 in the Appendix C.

Lemma 8.3. *Let S and S_∞ be the smooth step functions defined in Definition 8.2. It holds both S and S_∞ are monotone increasing functions and that $S(0) = 0$, $S(1) = 1$ and also $S'(0) = S'(1) = S''(0) = S''(1) = 0$. It also holds that $S_\infty(0) = 0$, $S_\infty(1) = 1$ and also $S_\infty^{(k)}(0) = S_\infty^{(k)}(1) = 0$ for every $k \in \mathbb{N}$. Additionally it holds for every x that $|S'(x)| \leq 2$, and $|S''(x)| \leq 6$ whereas $|S'_\infty(x)| \leq 16$, and $|S''_\infty(x)| \leq 32$.*



(a) Functions S and S_∞.

(b) The function P_3 from Example 8.4.

Figure 7: (a) The smooth step function S of order 1 and the smooth step function S_∞ of order ∞. As we can see both S and S_∞ are continuous and continuously differentiable functions but S_∞ is much more flat around 0 and 1 since it has all its derivatives equal to 0 both at the point 0 and at the point 1. This makes the S_∞ function infinitely many times differentiable. (b) The constructed P_3 function of the family of SEIC coefficients for single dimensional case with $N = 5$. For details we refer to the Example 8.4.

Proof. For the function S we compute $S'(x) = 30x^4 - 60x^3 + 30x^2$ for $x \in [0,1]$ and $S'(x) = 0$ for $x \notin [0,1]$. Therefore we can easily get that $|S'(x)| \leq 2$ for all $x \in \mathbb{R}$. We also have that $S''(x) = 120x^3 - 180x^2 + 60x$ for $x \in (0,1)$ and $S''(x) = 0$ for $x \notin [0,1]$ hence we can conclude that $|S''(x)| \leq 6$.

The calculations for S_∞ are more complicated. We have that

$$S'_\infty(x) = \ln(2) \frac{\exp\left(\frac{\ln(2)}{x(1-x)}\right)(1-2x(1-x))}{\left(\exp\left(\frac{\ln(2)}{x}\right) + \exp\left(\frac{\ln(2)}{1-x}\right)\right)^2 (1-x)^2 x^2}.$$

We set $h(x) \triangleq \left(\exp\left(\frac{\ln(2)}{x}\right) + \exp\left(\frac{\ln(2)}{1-x}\right)\right)(1-x)^2 x^2$ for $x \in [0,1]$ and doing simple calculations we get that for $x \leq 1/2$ it holds that $h(x) \geq \frac{1}{4}\exp\left(\frac{\ln(2)}{x}\right)x^2$. But the later can be easily lower bounded by $1/4$. Applying the same argument for $x \geq 1/2$ we get that in general $h(x) \geq 1/4$. Also it is not hard to see that for $x \leq 1/2$ it holds that $\exp\left(\frac{\ln(2)}{x(1-x)}\right) \leq 4\exp\left(\frac{\ln(2)}{x}\right)$, whereas for $x \geq 1/2$ it holds that $\exp\left(\frac{\ln(2)}{x(1-x)}\right) \leq 4\exp\left(\frac{\ln(2)}{1-x}\right)$. Combining all these we can conclude that $|S'_\infty(x)| \leq 16$. Using similar argument we can prove that $|S''_\infty(x)| \leq 32$. For all the derivatives of S_∞ we can inductively prove that

$$S_\infty^{(k)}(x) = \sum_{i=0}^{k-1} h_i(x) \cdot S_\infty^{(i)}(x),$$

where $h_0(1) = 0$ and all the functions $h_i(x)$ are bounded. Then the fact that all the derivatives of S_∞ vanish at 0 and at 1 follows by a simple inductive argument. □

Example 8.4 (Single Dimensional Smooth and Efficient Interpolation Coefficients). Using the smooth step functions that we described above we can get a construction of SEIC coefficients for

the single dimensional case. Unfortunately the extension to multiple dimensions is substantially harder and invokes new ideas that we explore later in this section. For the single dimensional problem of this example we have the interval $[0,1]$ divided with N discrete points and our goal is to design N functions P_1 - P_N that satisfy the properties (A) - (D) of Definition 7.3. A simple construction of such functions is the following

$$\mathsf{P}_i(x) = \begin{cases} S_\infty\left(N \cdot x - (i-1)\right) & x \leq \frac{i}{N-1} \\ S_\infty\left(i+1-N \cdot x\right) & x > \frac{i}{N-1} \end{cases}.$$

Based on Lemma 8.3 it is not hard then to see that P_i is twice differentiable and it has bounded first and second derivatives, hence it satisfies property (A) of Definition 8. Using the fact that $1 - S_\infty(x) = S_\infty(1-x)$ we can also prove property (B). Finally properties (C) and (D) can be proved via the definition of the coefficient P_i from above. In Figure 7 we can see the plot of P_3 for $N = 5$. We leave the exact proofs of this example as an exercise for the reader.

8.2 Construction of SEIC Coefficients in High-Dimensions

The goal of this section is to present the construction of the family $\mathcal{I}_{d,N}$ of smooth and efficient interpolation coefficients for every number of dimensions d and any discretization parameter N. Before diving into the details of our construction observe that even the 2-dimensional case with $N = 2$ is not trivial. In particular, the first attempt would be to define the SEIC coefficients based on the simple split of the square $[0,1]^2$ to two triangles divided by the diagonal of $[0,1]^2$. Then using any soft-max function that is twice continuously differentiable we define a convex combination at every triangle. Unfortunately this approach cannot work since the resulting coefficients have discontinuous gradients along the diagonal of $[0,1]^2$. We leave the presice calculations of this example as an exercise to the reader.

We start with some definitions about the orientation and the representation of the cubelets of the grid $([N]-1)^d$. Then we proceed with the definition of the Q_v functions in Definition 8.7. Finally using Q_v we can proceed with the construction of the SEIC coefficients.

Definition 8.5 (Source and Target of Cubelets)**.** Each cubelet $\left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$, where $\boldsymbol{c} \in ([N-1]-1)^d$ admits a **source vertex** $\boldsymbol{s^c} = (s_1, \ldots, s_d) \in ([N]-1)^d$ and a **target vertex** $\boldsymbol{t^c} = (t_1, \ldots, t_d) \in ([N]-1)^d$ defined as follows,

$$s_j = \begin{cases} c_j + 1 & c_j \text{ is odd} \\ c_j & c_j \text{ is even} \end{cases} \quad \text{and} \quad t_j = \begin{cases} c_j & c_j \text{ is odd} \\ c_j + 1 & c_j \text{ is even} \end{cases}$$

Notice that the source $\boldsymbol{s^c}$ and the target $\boldsymbol{t^c}$ are vertices of the cubelet whose down-left corner is $\boldsymbol{c}$.

Definition 8.6. (Canonical Representation) Let $\boldsymbol{x} \in [0,1]^d$ and $R(\boldsymbol{x}) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$ where $\boldsymbol{c} \in ([N-1]-1)^d$. The ***canonical representation* of $\boldsymbol{x}$ under cubelet with down-left corner $\boldsymbol{c}$**, denoted by $\boldsymbol{p_x^c} = (p_1, \ldots, p_d)$ is defined as follows,

$$p_j = \frac{x_j - s_j}{t_j - s_j}$$

where $\boldsymbol{t^c} = (t_1, \ldots, t_d)$ and $\boldsymbol{s^c} = (s_1, \ldots, s_d)$ are respectively the *target* and the *source* of $R(\boldsymbol{x})$.

Definition 8.7 (Defining the functions $Q_v(\boldsymbol{x})$)**.** Let $\boldsymbol{x} \in [0,1]^d$ lying in the cublet

$$R(\boldsymbol{x}) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right],$$

with corners $R_c(\boldsymbol{x}) = \{c_1, c_1+1\} \times \cdots \times \{c_d, c_d+1\}$, where $\boldsymbol{c} \in ([N-1]-1)^d$. Let also $\boldsymbol{s^c} = (s_1, \ldots, s_d)$ be the source vertex of $R(\boldsymbol{x})$ and $\boldsymbol{p_x^c} = (p_1, \ldots, p_d)$ be the canonical representation of $\boldsymbol{x}$. Then for each vertex $\boldsymbol{v} \in R_c(\boldsymbol{x})$ we define the following partition of the set of coordinates $[d]$,

$$A_v^c = \{j : \ |v_j - s_j| = 0\} \text{ and } B_v^c = \{j : \ |v_j - s_j| = 1\}$$

If there exist $j \in A_v^c$ and $\ell \in B_v^c$ such that $p_j \geq p_\ell$ then $Q_v^c(\boldsymbol{x}) = 0$. Otherwise we define[7]

$$Q_v^c(\boldsymbol{x}) = \begin{cases} \prod_{j \in A_v^c} \prod_{\ell \in B_v^c} S_\infty(S(p_\ell) - S(p_j)) & A_v^c, B_v^c \neq \varnothing \\ \prod_{\ell=1}^d S_\infty(1 - S(p_\ell)) & B_v^c = \varnothing \\ \prod_{j=1}^d S_\infty(S(p_j)) & A_v^c = \varnothing \end{cases}$$

where $S_\infty(x)$ and $S(x)$ are the smooth step function defined in Definition 8.2.

To provide a better understanding of the Definitions 8.5, 8.6, and 8.7 we present the following 3-dimensional example.

Example 8.8. We consider a case where $d = 3$ and $N = 3$. Let $\boldsymbol{x} = (1.3/3, 2.5/3, 0.3/3)$ lying in the cubelet $R(\boldsymbol{x}) = \left[\frac{1}{3}, \frac{2}{3}\right] \times \left[\frac{2}{3}, 1\right] \times \left[0, \frac{1}{3}\right]$, and let $\boldsymbol{c} = (1,2,0)$. Then the source of $R(\boldsymbol{x})$ is $\boldsymbol{s^c} = (2,2,0)$ and the target $\boldsymbol{t^c} = (1,3,1)$ (Definition 8.5). The canonical representation of $\boldsymbol{x}$ is $\boldsymbol{p_x^c} = (0.7, 0.5, 0.3)$ (Definition 8.6). The only vertices with no-zero coefficients $Q_v^c(\boldsymbol{x})$ are those belonging in the set $R_+(\boldsymbol{x}) = \{(1,3,1), (1,3,0), (1,2,0), (2,2,0)\}$ and again by Definition 8.7 we have that

- $Q_{(1,3,1)}(\boldsymbol{x}) = S_\infty(S(0.3)) \cdot S_\infty(S(0.5)) \cdot S_\infty(S(0.7))$,
- $Q_{(1,3,0)}(\boldsymbol{x}) = S_\infty(S(0.5) - S(0.3)) \cdot S_\infty(S(0.7) - S(0.3))$,
- $Q_{(1,2,0)}(\boldsymbol{x}) = S_\infty(S(0.7) - S(0.3)) \cdot S_\infty(S(0.7) - S(0.5))$,
- $Q_{(2,2,0)}(\boldsymbol{x}) = S_\infty(1 - S(0.3)) \cdot S_\infty(1 - S(0.5)) \cdot S_\infty(1 - S(0.7))$.

Now based on the Definitions 8.5, 8.6, and 8.7 we are ready to present the construction of the smooth and efficient interpolation coefficients.

Definition 8.9 (Construction of SEIC Coefficients)**.** Let $\boldsymbol{x} \in [0,1]^d$ lying in the cubelet $R(\boldsymbol{x}) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$. Then for each vertex $\boldsymbol{v} \in ([N]-1)^d$ the coefficient $\mathsf{P}_v(\boldsymbol{x})$ is defined as follows,

$$\mathsf{P}_v(\boldsymbol{x}) = \begin{cases} Q_v^c(\boldsymbol{x}) / (\sum_{v \in R_c(\boldsymbol{x})} Q_v^c(\boldsymbol{x})) & \text{if } \boldsymbol{v} \in R_c(\boldsymbol{x}) \\ 0 & \text{if } \boldsymbol{v} \notin R_c(\boldsymbol{x}) \end{cases}$$

where the functions $Q_v^c(\boldsymbol{x}) \geq 0$ are defined in Definition 8.7 for any $\boldsymbol{v} \in R_c(\boldsymbol{x})$.

[7] We note that in the following expression $\prod$ denotes the product symbol and should not be confused with the projection operator used in the previous sections.

8.3 Sketch of the Proof of Theorem 8.1

First it is necessary to argue that $\mathsf{P}_v(\boldsymbol{x})$ is a continuous function since it could be the case that $Q_v^c(\boldsymbol{x})/(\sum_{v\in R_c(\boldsymbol{x})} Q_v^c(\boldsymbol{x})) \neq Q_v^{c'}(\boldsymbol{x})/(\sum_{v\in V_{c'}} Q_v^{c'}(\boldsymbol{x}))$ for some point $\boldsymbol{x}$ that lies in the boundary of two adjacent cubelets with down-left corners $\boldsymbol{c}$ and $\boldsymbol{c}'$ respectively. We specifically design the coefficients $Q_v^c(\boldsymbol{x})$ such as the latter does not occur and this is the main reason that the definition of the function $Q_v^c(\boldsymbol{x})$ is slightly complicated. For this reason we prove the following lemma.

Lemma 8.10. *For any vertex $\boldsymbol{v} \in ([N]-1)^d$, $\mathsf{P}_v(\boldsymbol{x})$ is a continuous and twice differentiable function and for any $\boldsymbol{v} \notin R_c(\boldsymbol{x})$ it holds that $\mathsf{P}_v(\boldsymbol{x}) = \nabla\mathsf{P}_v(\boldsymbol{x}) = \nabla^2\mathsf{P}_v(\boldsymbol{x}) = 0$. Moreover, for every $\boldsymbol{x} \in [0,1]^d$ the set $R_+(\boldsymbol{x})$ of vertices $\boldsymbol{v} \in ([N]-1)^d$ such that $\mathsf{P}_v(\boldsymbol{x}) > 0$ satisfies $|R_+(\boldsymbol{x})| = d+1$.*

Based on Lemma 8.10 and the expression of P_v we can prove that the P_v coefficients defined in Definition 8.9 satisfy the properties (B) and (C) of the definition 7.3. To prove the properties (A) and (D) we also need the following two lemmas.

Lemma 8.11. *For any vertex $\boldsymbol{v} \in ([N]-1)^d$, it holds that*

1. $\left|\frac{\partial \mathsf{P}_v(\boldsymbol{x})}{\partial x_i}\right| \leq \Theta(d^{12}/\delta)$,
2. $\left|\frac{\partial^2 \mathsf{P}_v(\boldsymbol{x})}{\partial x_i\, \partial x_j}\right| \leq \Theta(d^{24}/\delta^2)$.

Lemma 8.12. *Let a point $\boldsymbol{x} \in [0,1]^d$ and $R_+(\boldsymbol{x})$ the set of vertices with $\mathsf{P}_v(\boldsymbol{x}) > 0$, then we have that*

1. *If $0 \leq x_i < 1/(N-1)$ then there always exists a vertex $\boldsymbol{v} \in R_+(\boldsymbol{x})$ such that $v_i = 0$.*
2. *If $1 - 1/(N-1) < x_i \leq 1$ then there always exists a vertex $\boldsymbol{v} \in R_+(\boldsymbol{x})$ such that $v_i = 1$.*

The proofs of Lemmas 8.10, 8.11, and 8.12 can be found in Appendix C. Based on Lemmas 8.10, 8.11, and 8.12 we are now ready to prove Theorem 8.1.

Proof of Theorem 8.1. The fact that the coefficients P_v satisfy the property (A) follows directly from Lemma 8.11. Property (B) follows directly from the definition of P_v in Definition 8.9 and the simple fact that $Q_v^c(\boldsymbol{x}) \geq 0$. Property (C) follows from the second part of Lemma 8.10. Finally Property (D) follows directly from Lemma 8.12. □

9 Unconditional Black-Box Lower Bounds

In this section our goal is to prove Theorem 4.5 based on the Theorem 4.4 that we proved in Section 7 and the known black box lower bounds that we know for PPAD by [HPV89]. In this section we assume that all the real number operation are performed with infinite precision.

Theorem 9.1 ([HPV89])**.** *Assume that there exists an algorithm A that has black-box oracle access to the value of a function $M : [0,1]^d \to [0,1]^d$ and outputs $\boldsymbol{w}^\star \in [0,1]^d$. There exists a universal constant $c > 0$ such that if M is 2-Lipschitz and $\|M(\boldsymbol{w}^\star) - \boldsymbol{w}^\star\|_2 \leq 1/(2c)$, then A has to make at least 2^d different oracle calls to the function value of M.*

It is easy to observe in the reduction in the proof of Theorem 7.2 is a black-box reduction and in every evaluation of the constructed circuit $\mathcal{C}_l$ only requires one evaluation of the input function M. Therefore the proof of Theorem 7.2 together with the Theorem 9.1 imply the following corollary.

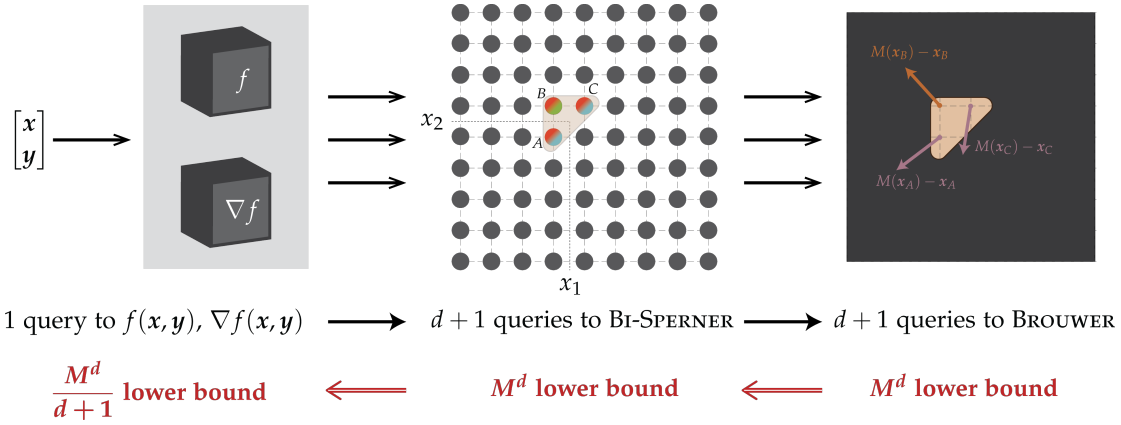


Figure 8: Pictorial representation on the way the black box lower bound follows from the white box PPAD-completeness that presented in Section 7 and the known black box lower bounds for the Brouwer problem by [HPV89]. In the figure we can see the four dimensional case of Section 6 that corresponds to the 2D-BiSperner and the 2-dimensional Brouwer. As we can see, in that case 1 query to $\mathcal{O}_f$ can be implemented with 3 queries to 2D-BiSperner and each of these can be implemented with 1 query to 2-dimensional Brouwer. In the high dimensional setting of Section 7, every query $(\boldsymbol{x}, \boldsymbol{y})$ to the oracle $\mathcal{O}_f$ to return the values $f(\boldsymbol{x}, \boldsymbol{y})$ and $\nabla f(\boldsymbol{x}, \boldsymbol{y})$ can be implemented via $d+1$ oracles to an HighD-BiSperner instance. Each of these oracles to HighD-BiSperner can be implemented via 1 oracle to a Brouwer instance. Therefore an M^d query lower bound for Brouwer implies an M^d query lower bound for HighD-BiSperner which in turn implies an $M^d/(d+1)$ query lower bound for our GDAFixedPoint and LR-LocalMinMax problems.

Corollary 9.2 (Black-Box Lower Bound for Bi-Sperner)**.** *Let $\mathcal{C}_l : ([N]-1)^d \to \{-1,1\}^d$ be an instance of the* HighD-BiSperner *problem with $N = O(d)$. Then any algorithm that has black-box oracle access to $\mathcal{C}_l$ and outputs a solution to the corresponding* HighD-BiSperner *problem, needs 2^d different oracle calls to the value of $\mathcal{C}_l$.*

Based on Corollary 9.2 and the reduction that we presented in Section 7, we are now ready to prove Theorem 4.5.

Proof of Theorem 4.5. This proof follows the steps of Figure 8. The last part of that figure is established in Corollary 9.2. So what is left to prove Theorem 4.5 is that for every instance of HighD-BiSperner we can construct a function f such that the oracle $\mathcal{O}_f$ can be implemented via $d+1$ queries to the instance of HighD-BiSperner and also every solution of GDAFixedPoint with oracle access $\mathcal{O}_f$ to f and ∇f reveals one solution of the starting HighD-BiSperner instance.

To construct this oracle $\mathcal{O}_f$ we follow exactly the reduction that we described in Section 7. The correctness of the reduction that we provide in Section 7 suffices to prove that every solution of the GDAFixedPoint with oracle access $\mathcal{O}_f$ to f and ∇f gives a solution to the initial HighD-BiSperner instance. So the only thing that remains is to bound the number of queries to the HighD-BiSperner instance that we need in order to implement the oracle $\mathcal{O}_f$. To do this consider the following definition of f based on an instance $\mathcal{C}_l$ of HighD-BiSperner from

Definition 7.4 with a scaling factor to make sure that the range of the function is $[-1,1]$

$$f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{d}\cdot\sum_{j=1}^{d}(x_j - y_j)\cdot\alpha_j(\boldsymbol{x})$$

where $\alpha_j(\boldsymbol{x}) = -\sum_{\boldsymbol{v}\in([N]-1)^d}\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})\cdot\mathcal{C}_l^j(\boldsymbol{v})$, and $\mathsf{P}_{\boldsymbol{v}}$ are the coefficients defined in Definition 7.3. From the property (C) of the coefficients $\mathsf{P}_{\boldsymbol{v}}$ we have that to evaluate $a_j(\boldsymbol{x})$ we only need the values $\mathcal{C}_l^j(\boldsymbol{v})$ for $d+1$ coefficients $\boldsymbol{v}$ and the same coefficients are needed to evaluate $a_j(\boldsymbol{x})$ for every j. This implies that for every $(\boldsymbol{x},\boldsymbol{y})$ we need $d+1$ oracle calls to the instance $\mathcal{C}_l$ of HighD-BiSperner so that $\mathcal{O}_f$ returns the value of $f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})$. If we take the gradient of $f_{\mathcal{C}_l}$ with respect to $(\boldsymbol{x},\boldsymbol{y})$ then an identical argument implies that the same set of $d+1$ queries to HighD-BiSperner are needed so that $\mathcal{O}_f$ returns the value of $\nabla f_{\mathcal{C}_l}(\boldsymbol{x},\boldsymbol{y})$ too. Therefore every query to the oracle $\mathcal{O}_f$ can be implemented via $d+1$ queries to $\mathcal{C}_l$. Now we can use Corollary 9.2 to get that the number of queries that we need in order to solve the GDAFixedPoint with oracle access $\mathcal{O}_f$ to f and ∇f is at least $2^d/(d+1)$. Finally observe that the proof of the Theorem 5.1 applies in th the black box model too. Hence finding solution of GDAFixedPoint in when we have black box access $\mathcal{O}_f$ to f and ∇f is equivalent to finding solutions of LR-LocalMinMax when we have exactly the same black box access $\mathcal{O}_f$ to f and ∇f. Therefore to find solutions of LR-LocalMinMax with black box access $\mathcal{O}_f$ to f and ∇f we need at least $2^d/(d+1)$ queries to $\mathcal{O}_f$ and the theorem follows by observing that in our proof the only parameters that depend on d are L, G, ε, and possibly δ but $1/\delta = O(\sqrt{L/\varepsilon})$ and hence the dependence of δ can be replaced by dependence on L and ε. □

10 Hardness in the Global Regime

In this section our goal is to prove that the complexity of the problems LocalMinMax and LocalMin is significantly increased when ε, δ lie outside the local regime, in the global regime. We start with the following theorem where we show that FNP-hardness of LocalMinMax.

Theorem 10.1. LocalMinMax *is* FNP*-hard even when* ε *is set to any value* $\leq 1/384$, δ *is set to any value* ≥ 1, *and even when* $\mathcal{P}(\boldsymbol{A},\boldsymbol{b}) = [0,1]^d$, $G = \sqrt{d}$, $L = d$, *and* $B = d$.

Proof. We now present a reduction from 3-SAT(3) to LocalMinMax that proves Theorem 10.1. First we remind the definition of the problem 3-SAT(3).

> **3-SAT(3).**
> Input: A boolean CNF-formula ϕ with boolean variables $x_1,\ldots,x_n$ such that every clause of ϕ has at most 3 boolean variables and every boolean variable appears to at most 3 clauses.
> Output: An assignment $\boldsymbol{x}\in\{0,1\}^n$ that satisfies ϕ, or $\perp$ if no such assignment exists.

Given an instance of 3-SAT(3) we first construct a polynomial $P_j(\boldsymbol{x})$ for each clause ϕ_j as follows: for each boolean variable x_i (there are n boolean variables x_i) we correspond a respective real-valued variable x_i. Then for each clause ϕ_j (there are m such clauses), let ℓ_i,ℓ_k,ℓ_m denote the literals participating in ϕ_j, $P_j(\boldsymbol{x}) = P_{ji}(\boldsymbol{x})\cdot P_{jk}(\boldsymbol{x})\cdot P_{jm}(\boldsymbol{x})$ where

$$P_{ji}(\boldsymbol{x}) = \begin{cases} 1-x_i & \text{if } \ell_i = x_i \\ x_i & \text{if } \ell_i = \overline{x_i} \end{cases}$$

Then the overall constructed function is

$$f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{z}) = \sum_{j=1}^{m} P_j(\boldsymbol{x}) \cdot (w_j - z_j)^2$$

where each w_j, z_j are additional variables associated with clause ϕ_j. The player that wants to minimize f controls $\boldsymbol{x}, \boldsymbol{w}$ vectors while the maximizing player controls the $\boldsymbol{z}$ variables.

Lemma 10.2. *The formula ϕ admits a satisfying assignment if and only if there exist an (ε, δ)-local min-max equilibrium of $f(\boldsymbol{x}, \boldsymbol{w})$ with $\varepsilon \leq 1/384$, $\delta = 1$ and $(\boldsymbol{x}, \boldsymbol{w}) \in [0,1]^{n+2m}$.*

Proof. Let us assume that there exists a satisfying assignment. Given such a satisfying assignment we will construct $((\boldsymbol{x}^\star, \boldsymbol{w}^\star), \boldsymbol{z}^\star)$ that is a $(0,1)$-local min-max equilibrium of f. We set each variable $x_i^\star \triangleq 1$ if and only if the respective boolean variable is true. Observe that this implies that $P_j(\boldsymbol{x}^\star) = 0$ for all j, meaning that the strategy profile $((\boldsymbol{x}^\star, \boldsymbol{w}^\star), \boldsymbol{z}^\star)$ is a global Nash equilibrium no matter the values of $\boldsymbol{w}^\star, \boldsymbol{z}^\star$.

On the opposite direction, let us assume that there exists an (ε, δ)-local min-max equilibrium of f with $\varepsilon = 1/384$ and $\delta = 1$. In this case we first prove that for each $j = 1, \ldots, m$

$$P_j(\boldsymbol{x}^\star) \leq 16 \cdot \varepsilon.$$

Fix any clause j. In case $\left|w_j^\star - z_j^\star\right| \geq 1/4$ then the minimizing player can further decrease f by at least $P_j(x)/16$ by setting $w_j^\star \triangleq z_j^\star$. On the other hand in case $\left|w_j^\star - z_j^\star\right| \leq 1/4$ then the maximizing player can increase f by at least $P_j(x^\star)/16$ by moving $z_j^\star$ either to 0 or to 1. We remark that both of the options are feasible since $\delta = 1$.

Now consider the probability distribution over the boolean assignments where each boolean variable x_i is independently selected to be true with probability $x_i^\star$. Then,

$$\mathbb{P}\left(\text{clause } \phi_j \text{ is not satisfied}\right) = P_j(\boldsymbol{x}^\star) \leq 16 \cdot \varepsilon = 1/24$$

Since each ϕ_j shares variables with at most 6 other clauses, the event of ϕ_j not being satisfied is dependent with at most 6 other events. By the Lovász Local Lemma [EL73], we get that the probability none of these events occur is positive. As a result, there exists a satisfying assignment. □

Hence the formula ϕ is satisfiable if and only if f has a $(1/384, 1)$-local min-max equilibrium point. What is left to prove the FNP-hardness is to show how we can find a satisfying assignment of ϕ given an approximate stationary point of f. This can be done using the celebrated results that provide constructive proofs of the Lovász Local Lemma [Mos09, MT10]. Finally to conclude the proof observe that since the f that we construct is a polynomial of degree 6 which can efficiently be described as a sum of monomials, we can trivially construct a Turing machine that computes the values of both f and ∇f in the polynomial time in the requested number of bits accuracy. The constructed function f is $\sqrt{d}$-Lipschitz and d-smooth, where d is the number of variables that is equal to $n + 2m$. More precisely since each variable x_i participates in at most 3 clauses, the real-valued variable x_i appears in at most 3 monomials P_j. Thus $-3 \leq \frac{\partial f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{x})}{\partial x_i} \leq 3$. Similarly it is not hard to see that $-2 \leq \frac{\partial f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{x})}{\partial w_j}, \frac{\partial f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{x})}{\partial z_j} \leq 2$. All the latter imply that $\|\nabla f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{z})\|_2 \leq \Theta(\sqrt{n+m})$, meaning that $f(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{z})$ is $\Theta(n+m)$-Lipschitz.

Using again the fact that each x_i participates in at most 3 monomials $P_j(\boldsymbol{x})$, we get that all the terms $\frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})}{\partial^2 x_i}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})}{\partial^2 w_j}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})}{\partial^2 z_j}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})}{\partial x_i \partial w_j}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})}{\partial x_i \partial z_j}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})}{\partial w_j \partial z_j} \in [-6,6]$. Thus the absolute value of each entry of $\nabla^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})$ is bounded by 6 and thus $\left\|\nabla^2 f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})\right\|_2 \leq \Theta(n+m)$, which implies the $\Theta(n+m)$-smoothness. Therefore our reduction produces a valid instance of LocalMinMax and hence the theorem follows. □

Next we show the FNP-hardness of LocalMin. As we can see there is a gap between Theorem 10.1 and Theorem 10.3. In particular, the FNP-hardness result of LocalMinMax is stronger since it holds for any $\delta \geq 1$ whereas for the FNP-hardness of LocalMin our proof needs $\delta \geq \sqrt{d}$ when the rest of the parameters remain the same.

Theorem 10.3. LocalMin *is* FNP*-hard even when* ε *is set to any value* $\leq 1/24$*,* δ *is set to any value* $\geq \sqrt{d}$*, and even when* $\mathcal{P}(\boldsymbol{A},\boldsymbol{b}) = [0,1]^d$*,* $G = \sqrt{d}$*,* $L = d$*, and* $B = d$*.*

Proof. We follow the same proof as in the proof of Theorem 10.1 but we instead set $f(\boldsymbol{x}) = \sum_{j=1}^{m} P_j(\boldsymbol{x})$ where $\boldsymbol{x} \in [0,1]^n$ (the number of variables is $d := n$). We then get that if the initial formula is satisfiable then there exist $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{A},\boldsymbol{b})$, such that $f(\boldsymbol{x}) = 0$. On the other hand if there exist $\boldsymbol{x} \in \mathcal{P}(\boldsymbol{A},\boldsymbol{b})$ such that $f(\boldsymbol{x}) \leq 1/24$ then the formula is satisfiable due to the Lovász Local Lemma [EL73]. Therefore the FNP-hardness follows again from the constructive proof of the Lovász Local Lemma [Mos09, MT10]. Setting $\delta \geq \sqrt{n}$ which equals the diameter of the feasibility set implies that in case there exists $\hat{\boldsymbol{x}}$ with $f(\hat{\boldsymbol{x}}) = 0$ then all (ε,δ)-LocalMin $\boldsymbol{x}^*$ must admit value $f(\boldsymbol{x}^*) \leq 1/24$ and thus a satisfying assignment is implied. □

Next we prove a black box lower bound for minimization in the global regime. The proof of following lower bound illustrates the strength of the SEIC coefficients presented in Section 8. The next Theorem can also be used to prove the FNP-hardness of LocalMin in the global regime but with worse Lipschitzness and smoothness parameters than the once at Theorem 10.3 and for this reason we present both of them.

Theorem 10.4. *In the worst case,* $\Omega\left(2^d/d\right)$ *value/gradient black-box queries are needed to determine a* (ε,δ)-LocalMin *for functions* $f(\boldsymbol{x}) : [0,1]^d \to [0,1]$ *with* $G = \Theta(d^{15})$*,* $L = \Theta(d^{22})$*,* $\varepsilon < 1$*,* $\delta = \sqrt{d}$*.*

Proof. The proof is based on the fact that given just *black-box access* to a boolean formula $\phi : \{0,1\}^d \mapsto \{0,1\}$, at least $\Omega(2^d)$ queries are needed in order to determine whether ϕ admits a satisfying assignment. The term *black-box access* refers to the fact that the clauses of the formula are not given and the only way to determine whether a specific boolean assignment is satisfying is by quering the specific binary string.

Given such a black-box oracle for a satisfying assignment d, we construct the function $f_\phi(\boldsymbol{x}) : [0,1]^d \mapsto [0,1]$ as follows:

1. for each corner $\boldsymbol{v} \in V$ of the $[0,1]^d$ hypercube, i.e. $\boldsymbol{v} \in \{0,1\}^d$, we set $f_\phi(\boldsymbol{v}) := 1 - \phi(\boldsymbol{v})$.
2. for the rest of the points $\boldsymbol{x} \in [0,1]^d/V$, $f_\phi(\boldsymbol{x}) := \sum_{\boldsymbol{v}\in V} P_{\boldsymbol{v}}(\boldsymbol{x}) \cdot f_\phi(\boldsymbol{v})$ where $P_{\boldsymbol{v}}$ are the coefficients of Definition 8.9.

We remind that by Lemma 8.11, we get that $\left\|\nabla f_\phi(\boldsymbol{x})\right\|_2 \leq \Theta(d^{12})$ and $\left\|\nabla^2 f_\phi(\boldsymbol{x})\right\|_2 \leq \Theta(d^{25})$, meaning that $f_\phi(\cdot)$ is $\Theta(d^{12})$-Lipschitz and $\Theta(d^{25})$-smooth. Moreover by Lemma 8.7 , for any

$\boldsymbol{x} \in [0,1]^n$ the set $V(x) = \{\boldsymbol{v} \in V : P_{\boldsymbol{v}}(\boldsymbol{x}) \neq 0\}$ has cardinality at most $d+1$, while at the same time $\sum_{\boldsymbol{v} \in V} P_{\boldsymbol{v}}(\boldsymbol{x}) = 1$.

In case ϕ is not satisfiable then $f_\phi(\boldsymbol{x}) = 1$ for all $\boldsymbol{x} \in [0,1]^d$ since $f_\phi(\boldsymbol{v}) = 1$ for all $\boldsymbol{v} \in V$. In case there exists a satisfying assignment $\boldsymbol{v}^*$ then $f_\phi(\boldsymbol{v}^*) = 0$. Since $\delta \geq \sqrt{d}$ that is the diameter of $[0,1]^d$, any (ε, δ)-LocalMin $\boldsymbol{x}^*$ must have $f_\phi(\boldsymbol{x}) \leq \varepsilon < 1$. Since $f_\phi(\boldsymbol{x}^*) \triangleq \sum_{\boldsymbol{v} \in V(\boldsymbol{x}^*)} P_{\boldsymbol{v}}(\boldsymbol{x}^*) \cdot f_\phi(\boldsymbol{v}^*) < 1$, there exists at least one vertex $\hat{\boldsymbol{v}} \in V(\boldsymbol{x})$ with $f_\phi(\hat{\boldsymbol{v}}) = 0$, meaning that $\phi(\boldsymbol{v}^*) = 1$. As a result, given an (ε, δ)-LocalMin $\boldsymbol{x}^*$ with $f_\phi(\boldsymbol{x}^*) < 1$, we can find a satisfying $\hat{\boldsymbol{v}}$ by querying $\phi(\boldsymbol{v})$ for each vertex $\boldsymbol{v} \in V(\boldsymbol{x}^*)$. Since $|V(\boldsymbol{x}^*)| \leq d+1$, this will take at most $d+1$ additional queries.

Up next, we argue that in case an (ε, δ)-LocalMin could be determined with less than $O(2^d/d)$ value/gradient queries, then determining whether ϕ admits a satisfying assignment could be done with less that $O(2^d)$ queries on ϕ (the latter is obviously impossible). Notice that any value/gradient query both $f_\phi(\boldsymbol{x})$ and $\nabla f_\phi(\boldsymbol{x})$ can be computed by querying the value $f_\phi(\boldsymbol{v})$ of the vertices $\boldsymbol{v} \in V(\boldsymbol{x})$. Since $|V(\boldsymbol{x})| \leq d+1$, any value/gradient query of f_ϕ can be simulated by $d+1$ queries on ϕ. □

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

[AAZB+17] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding approximate local minima faster than gradient descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1195–1199, 2017.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 214–223, 2017.

[Adl13] Ilan Adler. The equivalence of linear programs and zero-sum games. *International Journal of Game Theory*, 42(1):165–177, 2013.

[ADLH19] Leonard Adolphs, Hadi Daneshmand, Aurelien Lucchi, and Thomas Hofmann. Local saddle point optimization: A curvature exploitation approach. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 486–495, 2019.

[ADSG19] Mohammad Alkousa, Darina Dvinskikh, Fedor Stonyakin, and Alexander Gasnikov. Accelerated methods for composite non-bilinear saddle point problem. *arXiv preprint arXiv:1906.03620*, 2019.

[ALW19] Jacob Abernethy, Kevin A Lai, and Andre Wibisono. Last-iterate convergence rates for min-max optimization. *arXiv preprint arXiv:1906.02027*, 2019.

[AMLJG20] Waïss Azizian, Ioannis Mitliagkas, Simon Lacoste-Julien, and Gauthier Gidel. A tight and unified analysis of extragradient for a whole spectrum of differentiable games. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[BCB12] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[BCE+95] Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 303–314, 1995.

[BIQ+17] Aleksandrs Belovs, Gábor Ivanyos, Youming Qiao, Miklos Santha, and Siyi Yang. On the polynomial parity argument complexity of the combinatorial nullstellensatz. In *Proceedings of the 32nd Computational Complexity Conference*, pages 1–24, 2017.

[Bla56] David Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.*, 6(1):1–8, 1956.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science, (FOCS)*, 2015.

[Bre76] Richard P Brent. Fast multiple-precision evaluation of elementary functions. *Journal of the ACM (JACM)*, 23(2):242–251, 1976.

[CBL06] Nikolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *Journal of the ACM (JACM)*, 56(3):1–57, 2009.

[CPY17] Xi Chen, Dimitris Paparas, and Mihalis Yannakakis. The complexity of non-monotone markets. *J. ACM*, 64(3):20:1–20:56, 2017.

[Dan51] George B. Dantzig. A proof of the equivalence of the programming problem and the game problem. In *Koopmans, T. C., editor(s), Activity Analysis of Production and Allocation*. Wiley, New York, 1951.

[Das13] Constantinos Daskalakis. On the complexity of approximating a nash equilibrium. *ACM Transactions on Algorithms (TALG)*, 9(3):1–35, 2013.

[Das18] Constantinos Daskalakis. Equilibria, Fixed Points, and Computational Complexity - Nevanlinna Prize Lecture. *Proceedings of the International Congress of Mathematicians (ICM)*, 1:147–209, 2018.

[DFS20] Argyrios Deligkas, John Fearnley, and Rahul Savani. Tree polymatrix games are ppad-hard. *CoRR*, abs/2002.12119, 2020.

[DGP09] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[DISZ18] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. In *International Conference on Learning Representations (ICLR 2018)*, 2018.

[DP11] Constantinos Daskalakis and Christos Papadimitriou. Continuous local search. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 790–804. SIAM, 2011.

[DP18] Constantinos Daskalakis and Ioannis Panageas. The limit points of (optimistic) gradient descent in min-max optimization. In *Advances in Neural Information Processing Systems*, pages 9236–9246, 2018.

[DP19] Constantinos Daskalakis and Ioannis Panageas. Last-iterate convergence: Zero-sum games and constrained min-max optimization. *Innovations in Theoretical Computer Science*, 2019.

[DTZ18] Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. A converse to banach's fixed point theorem and its CLS-completeness. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2018.

[EL73] Paul Erdős and László Lovász. 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, 1973.

[EY10] Kousha Etessami and Mihalis Yannakakis. On the complexity of nash equilibria and other fixed points. *SIAM Journal on Computing*, 39(6):2531–2597, 2010.

[FG18] Aris Filos-Ratsikas and Paul W. Goldberg. Consensus halving is ppa-complete. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2018.

[FG19] Aris Filos-Ratsikas and Paul W. Goldberg. The complexity of splitting necklaces and bisecting ham sandwiches. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2019.

[FP07] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007.

[FPT04] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, 2004.

[FRHSZ20a] Aris Filos-Ratsikas, Alexandros Hollender, Katerina Sotiraki, and Manolis Zampetakis. Consenus-halving: Does it ever get easier? *arXiv preprint arXiv:2002.11437*, 2020.

[FRHSZ20b] Aris Filos-Ratsikas, Alexandros Hollender, Katerina Sotiraki, and Manolis Zampetakis. A topological characterization of modulo-p arguments and implications for necklace splitting. *arXiv preprint arXiv:2003.11974*, 2020.

[GH19] Paul W. Goldberg and Alexandros Hollender. The hairy ball problem is ppad-complete. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2019.

[GHP+19] Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Rémi Le Priol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1802–1811, 2019.

[GKSZ19] Mika Göös, Pritish Kamath, Katerina Sotiraki, and Manolis Zampetakis. On the complexity of modulo-q arguments and the chevalley-warning theorem. *arXiv preprint arXiv:1912.04467*, 2019.

[Goo16] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[GPDO20] Noah Golowich, Sarath Pattathil, Constantinos Daskalakis, and Asuman E. Ozdaglar. Last iterate is slower than averaged iterate in smooth convex-concave saddle point problems. *CoRR*, abs/2002.00057, 2020.

[GPM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 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*, pages 2672–2680, 2014.

[HA18] Erfan Yazdandoost Hamedani and Necdet Serhat Aybat. A primal-dual algorithm for general convex-concave saddle point problems. *arXiv preprint arXiv:1803.01401*, 2018.

[Haz16] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[HPV89] M. D. Hirsch, C. H. Papadimitriou, and S. A. Vavasis. Exponential lower bounds for finding brouwer fixed points. *Journal of Complexity*, 5:379–416, 1989.

[Jeř16] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016.

[JGN+17] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1724–1732. JMLR. org, 2017.

[JNJ19] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? *arXiv preprint arXiv:1902.00618*, 2019.

[JPY88] David S Johnson, Christos H Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of computer and system sciences*, 37(1):79–100, 1988.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KM18] Pravesh K. Kothari and Ruta Mehta. Sum-of-squares meets Nash: lower bounds for finding any equilibrium. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2018.

[KM19] Weiwei Kong and Renato DC Monteiro. An accelerated inexact proximal point method for solving nonconvex-concave min-max problems. *arXiv preprint arXiv:1905.13433*, 2019.

[Kor76] GM Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

[LJJ19] Tianyi Lin, Chi Jin, and Michael I Jordan. On gradient descent ascent for nonconvex-concave minimax problems. *arXiv preprint arXiv:1906.00331*, 2019.

[LJJ20] Tianyi Lin, Chi Jin, and Michael Jordan. Near-optimal algorithms for minimax optimization. *arXiv preprint arXiv:2002.02417*, 2020.

[LPP+19] Jason D. Lee, Ioannis Panageas, Georgios Piliouras, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. First-order methods almost always avoid strict saddle points. *Math. Program.*, 176(1-2):311–337, 2019.

[LS19] Tengyuan Liang and James Stokes. Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 907–915, 2019.

[LTHC19] Songtao Lu, Ioannis Tsaknakis, Mingyi Hong, and Yongxin Chen. Hybrid block successive approximation for one-sided non-convex min-max problems: algorithms and applications. *arXiv preprint arXiv:1902.08294*, 2019.

[Meh14] Ruta Mehta. Constant rank bimatrix games are ppad-hard. In *Proceedings of the 46th Symposium on Theory of Computing (STOC)*, 2014.

[MGN18] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning*, pages 3481–3490, 2018.

[MMS+18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[MOP19] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. *arXiv preprint arXiv:1901.08511*, 2019.

[Mos09] Robin A Moser. A constructive proof of the lovász local lemma. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 343–350, 2009.

[MP89] N Meggido and CH Papadimitriou. A note on total functions, existence theorems, and computational complexity. Technical report, Tech. report, IBM, 1989.

[MPP18] Panayotis Mertikopoulos, Christos H. Papadimitriou, and Georgios Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2018.

[MPPSD16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[MR18] Eric Mazumdar and Lillian J Ratliff. On the convergence of gradient-based learning in continuous games. *arXiv preprint arXiv:1804.05464*, 2018.

[MSV20] Oren Mangoubi, Sushant Sachdeva, and Nisheeth K Vishnoi. A provably convergent and practical algorithm for min-max optimization with applications to gans. *arXiv preprint arXiv:2006.12376*, 2020.

[MT10] Robin A Moser and Gábor Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):1–15, 2010.

[MV20] Oren Mangoubi and Nisheeth K Vishnoi. A second-order equilibrium in nonconvex-nonconcave min-max optimization: Existence and algorithm. *arXiv preprint arXiv:2006.12363*, 2020.

[Nem04] Arkadi Nemirovski. Interior point polynomial time methods in convex programming. *Lecture notes*, 2004.

[NSH+19] Maher Nouiehed, Maziar Sanjabi, Tianjian Huang, Jason D Lee, and Meisam Razaviyayn. Solving a class of non-convex min-max games using iterative first order methods. In *Advances in Neural Information Processing Systems*, pages 14905–14916, 2019.

[NY83] Arkadiĭ Semenovich Nemirovsky and David Borisovich Yudin. *Problem complexity and method efficiency in optimization*. Chichester: Wiley, 1983.

[OX19] Yuyuan Ouyang and Yangyang Xu. Lower complexity bounds of first-order methods for convex-concave bilinear saddle-point problems. *Mathematical Programming*, pages 1–35, 2019.

[Pap94a] C Papadimitriou. *Computational Complexity*. Addison Welsey, 1994.

[Pap94b] Christos H Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and system Sciences*, 48(3):498–532, 1994.

[RKK18] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[RLLY18] Hassan Rafique, Mingrui Liu, Qihang Lin, and Tianbao Yang. Non-convex min-max optimization: Provable algorithms and applications in machine learning. *arXiv preprint arXiv:1810.02060*, 2018.

[Ros65] J Ben Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica: Journal of the Econometric Society*, pages 520–534, 1965.

[Rub15] Aviad Rubinstein. Inapproximability of nash equilibrium. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, 2015.

[Rub16] Aviad Rubinstein. Settling the complexity of computing approximate two-player nash equilibria. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 258–265. IEEE, 2016.

[SS12] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[SY91] Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM J. Comput.*, 20(1):56–87, 1991.

[SZZ18] Katerina Sotiraki, Manolis Zampetakis, and Giorgos Zirdelis. Ppp-completeness with connections to cryptography. In *Proceddings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2018.

[TJNO19] Kiran K Thekumparampil, Prateek Jain, Praneeth Netrapalli, and Sewoong Oh. Efficient algorithms for smooth minimax optimization. In *Advances in Neural Information Processing Systems*, pages 12659–12670, 2019.

[vN28] John von Neumann. Zur Theorie der Gesellschaftsspiele. In *Math. Ann.*, pages 295–320, 1928.

[VY11] Vijay V. Vazirani and Mihalis Yannakakis. Market equilibrium under separable, piecewise-linear, concave utilities. *J. ACM*, 58(3):10:1–10:25, 2011.

[WZB19] Yuanhao Wang, Guodong Zhang, and Jimmy Ba. On solving minimax optimization locally: A follow-the-ridge approach. In *International Conference on Learning Representations*, 2019.

[Zha19] Renbo Zhao. Optimal algorithms for stochastic three-composite convex-concave saddle point problems. *arXiv preprint arXiv:1903.01687*, 2019.

A Proof of Theorem 4.1

We first remind the definition of the 3-SAT(3) problem that we will use for our reduction.

3-SAT(3).
INPUT: A boolean CNF-formula ϕ with boolean variables $x_1, \dots, x_n$ such that every clause of ϕ has at most 3 boolean variables and every boolean variable appears to at most 3 clauses.
OUTPUT: An assignment $\boldsymbol{x} \in \{0,1\}^n$ that satisfies ϕ, or $\perp$ if no such assignment exists.

It is well known that 3-SAT(3) is FNP-complete, for details see §9.2 of [Pap94a]. To prove Theorem 4.1, we reduce 3-SAT(3) to ε-StationaryPoint.

Given an instance of 3-SAT(3) we construct the function $f : [0,1]^{n+m} \to [0,1]$, where m is the number of clauses of ϕ. For each literal x_i we assign a real-valued variable which by abuse of notation we also denote x_i and it would be clear from the context whether we refer to the literal or the real-valued variable. Then for each clause ϕ_j of ϕ, we construct a polynomial $P_j(x)$ as follows: if ℓ_i, ℓ_k, ℓ_m are the literals participating in ϕ_j, then $P_j(x) = P_{ji}(x) \cdot P_{jk}(x) \cdot P_{jm}(x)$ where

$$P_{ji}(x) = \begin{cases} 1 - x_i & \text{if } \ell_i = x_i \\ x_i & \text{if } \ell_i = \overline{x_i} \end{cases}$$

The overall constructed function is $f(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=1}^{m} w_j \cdot P_j(\boldsymbol{x})$, where each w_j is an additional variable associated with clause ϕ_j. Notice that $0 \leq \frac{\partial f(\boldsymbol{x},\boldsymbol{w})}{\partial w_j} \leq 1$ and $-3 \leq \frac{\partial f(\boldsymbol{x},\boldsymbol{w})}{\partial x_i} \leq 3$ since the boolean variable x_i participates in at most 3 clauses. As a result, $\|\nabla f(\boldsymbol{x}, \boldsymbol{w})\|_2 \leq \Theta(\sqrt{n+m})$, meaning that $f(\boldsymbol{x}, \boldsymbol{w})$ is G-Lipschitz with $G = \Theta(\sqrt{n+m})$. Also notice that all the entries of $\nabla^2 f(\boldsymbol{x}, \boldsymbol{w})$, i.e. $\frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w})}{\partial^2 x_i} = \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w})}{\partial^2 w_j}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w})}{\partial x_i \, \partial w_j}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w})}{\partial x_i \, \partial x_m}, \frac{\partial^2 f(\boldsymbol{x},\boldsymbol{w})}{\partial w_k \, \partial w_j} \in [-3,3]$. As a result, $\left\|\nabla^2 f(\boldsymbol{x}, \boldsymbol{w})\right\|_2 \leq \Theta(n+m)$, meaning that $f(\boldsymbol{x}, \boldsymbol{w})$ is L-smooth with $L = \Theta(n+m)$.

Lemma A.1. *There exists a satisfying assignment for the clauses $\phi_1, \dots, \phi_m$ if and only if there solution of the constructed* StationaryPoint *with $\varepsilon = 1/24$ a admits solution $(\boldsymbol{x}^\star, \boldsymbol{w}^\star) \in [0,1]^{n+m}$ such that $\|\nabla f(\boldsymbol{x}^\star, \boldsymbol{w}^\star)\|_2 < 1/24$.*

Proof. By the definition of StationaryPoint, in case there exists a pair of points $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}}) \in [0,1]^{n+m}$ with $\|\nabla f(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}})\|_2 < \varepsilon/2 = 1/48$, then a pair of points $(\boldsymbol{x}^\star, \boldsymbol{w}^\star)$ with $\|\nabla f(\boldsymbol{x}^\star, \boldsymbol{w}^\star)\|_2 < \varepsilon = 1/24$ must be returned. In case $\|\nabla f(\boldsymbol{x}, \boldsymbol{w})\|_2 > \varepsilon = 1/24$ for all $(\boldsymbol{x}, \boldsymbol{w}) \in [0,1]^{n+m}$, the null symbol $\perp$ is returned.

Let us assume that there exists a satisfying assignment of ϕ. Consider the solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}})$ constructed as follows: each variable $\hat{x}_i$ is set to 1 iff the respective boolean variable is true and $\hat{w}_j = 0$ for all $j = 1, \dots, m$. Since the assignment satisfies the CNF-formula ϕ, there exists at least one true literal in each clause ϕ_j which means that $P_j(x) = 0$ for all $j = 1, \dots, m$. As a result $\frac{\partial f(\hat{\boldsymbol{x}},\hat{\boldsymbol{w}})}{\partial w_j} = P_j(\hat{\boldsymbol{x}}) = 0$ for all $j = 1, \dots, m$. At the same time, $\frac{\partial f(\hat{\boldsymbol{x}},\hat{\boldsymbol{w}})}{\partial x_i} = 0$ since $\hat{w}_j = 0$ for all $j = 1, \dots, m$. Overall we have that $\nabla f(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}}) = 0 < 1/48 = \varepsilon/2$. As a result, the constructed StationaryPoint instance must return a solution $(\boldsymbol{x}^\star, \boldsymbol{w}^\star)$ with $\|\nabla f(\boldsymbol{x}^\star, \boldsymbol{w}^\star)\|_2 < \frac{1}{24} = \varepsilon$.

On the opposite direction, the existence of a pair of points $(\boldsymbol{x}^\star, \boldsymbol{w}^\star)$ with $\|\nabla f(\boldsymbol{x}^\star, \boldsymbol{w}^\star)\|_2 < 1/24$ implies $P_j(\boldsymbol{x}^*) < 1/24$ for all $j = 1 \dots m$. Consider the probability distribution over the boolean assignments in which *each boolean variable x_i is independently selected to be true with probability $x_i^\star$*. Then,

$$\mathbb{P}\left(\text{clause } \phi_j \text{ is not satisfied}\right) = P_j(\boldsymbol{x}^\star) < 1/24$$

Since ϕ_j shares variables with at most 6 other clauses, the bad event of ϕ_j not being satisfied is dependent with at most 6 other bad events. By Lovász Local Lemma [EL73], we get that the probability none of the events occurs is positive. As a result, there exists a satisfying assignment. □

Using Lemma A.1 we can conclude that ϕ is satisfiable if and only if f has a $1/24$-approximate stationary point. What is left to prove the FNP-hardness is to show how we can find a satisfying assignment of ϕ given an approximate stationary point of f. This can be done using the celebrated results that provide constructive proofs of the Lovász Local Lemma [Mos09, MT10]. Finally, we remind that the constructed function f is $\Theta\left(\sqrt{d}\right)$-Lipschitz and $\Theta(d)$-smooth, where d is the number of variables that is equal to $n+m$.

B Missing Proofs from Section 5

In this section we give proofs for the statements presented in Section 5. These statements establish the totality and inclusion to PPAD of LR-LocalMinMax and GDAFixedPoint.

B.1 Proof of Theorem 5.1

We start with establishing claim "1." in the statement of the theorem. It will be clear that our proof will provide a polynomial-time reduction from LR-LocalMinMax to GDAFixedPoint. Suppose that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is an α-approximate fixed point of F_{GDA}, where α is the specified in the theorem statement function of δ, G and L. To simplify our proof, we abuse notation and define $f(\boldsymbol{x}) \triangleq f(\boldsymbol{x}, \boldsymbol{y}^\star)$, $\nabla f(\boldsymbol{x}) \triangleq \nabla_x f(\boldsymbol{x}, \boldsymbol{y}^\star)$, $K \triangleq \{\boldsymbol{x} \mid (\boldsymbol{x}, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})\}$ and $\hat{\boldsymbol{x}} \triangleq \Pi_K(\boldsymbol{x}^\star - \nabla f(\boldsymbol{x}^\star))$. Because $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is an α-approximate fixed point of F_{FDA}, it follows that $\|\hat{\boldsymbol{x}} - \boldsymbol{x}^\star\|_2 < \alpha$.

Claim B.1. *$\langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x}^\star - \boldsymbol{x} \rangle < (G + \delta + \alpha) \cdot \alpha$, for all $\boldsymbol{x} \in K \cap B_{d_1}(\delta; \boldsymbol{x}^\star)$.*

Proof. Using the fact that $\hat{\boldsymbol{x}} = \Pi_K(\boldsymbol{x}^\star - \nabla f(\boldsymbol{x}^\star))$ and that K is a convex set we can apply Theorem 1.5.5 (b) of [FP07] to get that

$$\langle \boldsymbol{x}^\star - \nabla f(\boldsymbol{x}^\star) - \hat{\boldsymbol{x}}, \boldsymbol{x} - \hat{\boldsymbol{x}} \rangle \leq 0, \forall \boldsymbol{x} \in K. \tag{B.1}$$

Next, we do some simple algebra to get that, for all $\boldsymbol{x} \in K \cap B_{d_1}(\delta; \boldsymbol{x}^\star)$,

$$\begin{aligned}
\langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x}^\star - \boldsymbol{x} \rangle &= \langle \boldsymbol{x}^\star - \nabla f(\boldsymbol{x}^\star) - \hat{\boldsymbol{x}}, \boldsymbol{x} - \hat{\boldsymbol{x}} \rangle + \langle \boldsymbol{x} - \hat{\boldsymbol{x}} - \nabla f(\boldsymbol{x}^\star), \hat{\boldsymbol{x}} - \boldsymbol{x}^\star \rangle \\
&\overset{(B.1)}{\leq} \langle \boldsymbol{x} - \hat{\boldsymbol{x}} - \nabla f(\boldsymbol{x}^\star), \hat{\boldsymbol{x}} - \boldsymbol{x}^\star \rangle \\
&\leq (\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 + \|\nabla f(\boldsymbol{x}^\star)\|_2) \|\hat{\boldsymbol{x}} - \boldsymbol{x}^\star\|_2 < (G + \delta + \alpha) \cdot \alpha,
\end{aligned}$$

where the second to last inequality follows from Cauchy–Schwarz inequality and the triangle inequality, and the last inequality follows from the triangle inequality and the following facts: (1) $\|\boldsymbol{x}^\star - \hat{\boldsymbol{x}}\|_2 < \alpha$, (2) $\boldsymbol{x} \in B_{d_1}(\delta; \boldsymbol{x}^\star)$, and (3) $\|\nabla f(\boldsymbol{x}, \boldsymbol{y})\|_2 \leq G$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$. □

For all $\boldsymbol{x} \in K \cap B_{d_1}(\delta; \boldsymbol{x}^\star)$, from the L-smoothness of f we have that

$$|f(\boldsymbol{x}) - (f(\boldsymbol{x}^\star) + \langle \nabla f(\boldsymbol{x}^*), \boldsymbol{x} - \boldsymbol{x}^\star \rangle)| \leq \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2. \tag{B.2}$$

We distinguish two cases:

1. $f(x^\star) \leq f(x)$: In this case we stop, remembering that

$$f(x^\star) \leq f(x). \tag{B.3}$$

2. $f(x^\star) > f(x)$: In this case, we consider two further sub-cases:

   (a) $\langle \nabla f(x^*), x - x^\star \rangle \geq 0$: in this sub-case, Eq (B.2) gives

   $$f(x^\star) - f(x) + \langle \nabla f(x^*), x - x^\star \rangle \leq \frac{L}{2} \|x - x^\star\|_2^2$$

   Thus

   $$f(x^\star) \leq f(x) + \frac{L}{2} \|x - x^\star\|_2^2 \leq f(x) + \frac{L}{2}\delta^2 < f(x) + \varepsilon, \tag{B.4}$$

   where for the last inequality we used that $x \in B_{d_1}(\delta; x^\star)$, and that $\delta < \sqrt{2\varepsilon/L}$.

   (b) $\langle \nabla f(x^*), x - x^\star \rangle < 0$: in this sub-case, Eq (B.2) gives

   $$f(x^\star) - f(x) - \langle \nabla f(x^*), x^\star - x \rangle \leq \frac{L}{2} \|x - x^\star\|_2^2.$$

   Thus

   $$\begin{aligned} f(x^\star) &\leq f(x) + \langle \nabla f(x^*), x^\star - x \rangle + \frac{L}{2} \|x - x^\star\|_2^2 \\ &\leq f(x) + \langle \nabla f(x^*), x^\star - x \rangle + \frac{L}{2} \cdot \delta^2 \\ &< f(x) + (G + \delta + \alpha) \cdot \alpha + \frac{L}{2} \cdot \delta^2 \\ &\leq f(x) + \varepsilon, \end{aligned} \tag{B.5}$$

   where the second inequality follows from the fact that $x \in B_{d_1}(\delta; x^\star)$, the third inequality follows from Claim B.1, and the last inequality follows from the constraints $\delta < \sqrt{2\varepsilon/L}$ and $\alpha \leq \frac{\sqrt{(G+\delta)^2 + 4(\varepsilon - \frac{L}{2}\delta^2)} - (G+\delta)}{2}$.

In all cases, we get from (B.3), (B.4) and (B.5) that $f(x^\star) < f(x) + \varepsilon$, for all $x \in K \cap B_{d_1}(\delta; x^\star)$. Thus, lifting our abuse of notation, we get that $f(x^\star, y^\star) < f(x, y^\star) + \varepsilon$, for all $x \in \{x \mid x \in B_{d_1}(\delta; x^\star) \text{ and } (x, y^\star) \in \mathcal{P}(A, b)\}$. Using an identical argument we can also show that $f(x^\star, y^\star) > f(x^\star, y) - \varepsilon$ for all $y \in \{y \mid y \in B_{d_2}(\delta; y^\star) \text{ and } (x^\star, y) \in \mathcal{P}(A, b)\}$. The first part of the theorem follows.

Now let us establish claim "2." in the theorem statement. It will be clear that our proof will provide a polynomial-time reduction from GDAFixedPoint to LR-LocalMinMax. For the choice of parameters ε and δ described in the theorem statement, we will show that, if $(x^\star, y^\star)$ is an (ε, δ)-local min-max equilibrium of f, then $\|F_{GDAx}(x^\star, y^\star) - x^\star\|_2 < \alpha/2$ and $\left\|F_{GDAy}(x^\star, y^\star) - y^\star\right\|_2 < \alpha/2$. The second part of the theorem will then follow. We only prove that $\|F_{GDAx}(x^\star, y^\star) - x^\star\|_2 < \alpha/2$, as the argument for $y^\star$ is identical. In the argument below we abuse notation in the same way we described earlier. With that notation we will show that $\|\hat{x} - x^\star\|_2 < \alpha/2$.

Proof that $\|\hat{x} - x^\star\| < \alpha/2$. From our choice of ε and δ, it is easy to see that $\delta = \alpha/(5L + 2) < \alpha/2$. Thus, if $\|\hat{x} - x^\star\| < \delta$, then we automatically get $\|\hat{x} - x^\star\| < \alpha/2$. So it remains to handle

the case $\|\hat{\boldsymbol{x}} - \boldsymbol{x}^\star\| \geq \delta$. We choose $\boldsymbol{x}_c \triangleq \boldsymbol{x}^\star + \delta \frac{\hat{\boldsymbol{x}} - \boldsymbol{x}^\star}{\|\hat{\boldsymbol{x}} - \boldsymbol{x}^\star\|_2}$. It is easy to see that $\boldsymbol{x}_c \in B_{d_1}(\delta; \boldsymbol{x}^\star)$ and hence we get that

$$\begin{aligned} f(\boldsymbol{x}^\star) - \varepsilon < f(\boldsymbol{x}_c) &\leq f(\boldsymbol{x}^\star) + \langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x}_c - \boldsymbol{x}^\star \rangle + \frac{L}{2} \|\boldsymbol{x}_c - \boldsymbol{x}^\star\|^2 \\ &\leq f(\boldsymbol{x}^\star) + \langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x}_c - \boldsymbol{x}^\star \rangle + \frac{\varepsilon}{2}, \end{aligned}$$

where the first inequality follows from the fact that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is an (ε, δ)-local min-max equilibrium, the second inequality follows from the L-smoothness of f, and the third inequality follows from $\|\boldsymbol{x}_c - \boldsymbol{x}^\star\| \leq \delta$ and our choice of $\delta = \sqrt{\varepsilon / L}$. The above implies:

$$\langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x}^\star - \boldsymbol{x}_c \rangle < 3\varepsilon/2.$$

Since $\hat{\boldsymbol{x}} - \boldsymbol{x}^\star = (\boldsymbol{x}_c - \boldsymbol{x}^\star) \cdot \|\hat{\boldsymbol{x}} - \boldsymbol{x}^\star\|_2 / \delta$ we get that $\langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x}^\star - \hat{\boldsymbol{x}} \rangle < \frac{3\varepsilon}{2\delta} \|\boldsymbol{x}^\star - \hat{\boldsymbol{x}}\|_2$. Therefore

$$\begin{aligned} \|\boldsymbol{x}^\star - \hat{\boldsymbol{x}}\|_2^2 &= \langle \boldsymbol{x}^\star - \nabla f(\boldsymbol{x}^\star) - \hat{\boldsymbol{x}}, \boldsymbol{x}^\star - \hat{\boldsymbol{x}} \rangle + \langle \nabla f(\boldsymbol{x}^\star), \boldsymbol{x}^\star - \hat{\boldsymbol{x}} \rangle \\ &< \frac{3\varepsilon}{2\delta} \|\boldsymbol{x}^\star - \hat{\boldsymbol{x}}\|_2 \end{aligned}$$

where in the above inequality we have also used (B.1). As a result, $\|\boldsymbol{x}^\star - \hat{\boldsymbol{x}}\|_2 < \frac{3\varepsilon}{2\delta} < \alpha/2$.

B.2 Proof of Theorem 5.2

We provide a polynomial-time reduction from GDAFixedPoint to Brouwer. This establishes both the totality of GDAFixedPoint and its inclusion to PPAD, since Brouwer is both total and lies in PPAD, as per Lemma 2.5. It also establishes the totality and inclusion to PPAD of LR-LocalMinMax, since LR-LocalMinMax is polynomial-time reducible to GDAFixedPoint, as shown in Theorem 5.1.

We proceed to describe our reduction. Suppose that f is the G-Lipschitz and L-smooth function provided as input to GDAFixedPoint. Suppose also that α is the approximation parameter provided as input to GDAFixedPoint. Given f and α, we define function $M : \mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \to \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$, which serves as input to Brouwer, as follows:

$$M(\boldsymbol{x}, \boldsymbol{y}) = \Pi_{\mathcal{P}(\boldsymbol{A}, \boldsymbol{b})} \left[(\boldsymbol{x} - \nabla_x f(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{y} + \nabla_y f(\boldsymbol{x}, \boldsymbol{y})) \right].$$

Given that f is L-smooth, it follows that M is $(L+1)$-Lipschitz. We set the approximation parameter provided as input to Brouwer be $\gamma = \alpha^2 / 4(G + 2\sqrt{d})$.

To show the validity of the afore-described reduction, we prove that every feasible point $(\boldsymbol{x}^\star, \boldsymbol{y}^\star) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$ that is a γ-approximate fixed point of M, i.e. $\|M(\boldsymbol{x}^\star, \boldsymbol{y}^\star) - (\boldsymbol{x}^\star, \boldsymbol{y}^\star)\|_2 < \gamma$ is also an α-approximate fixed point of F_{GDA}. Observe that since $\mathcal{P}(\boldsymbol{A}, \boldsymbol{b}) \subseteq [0,1]^d$ it holds that $\|(\boldsymbol{x}, \boldsymbol{y}) - (\boldsymbol{x}', \boldsymbol{y}')\|_2 \leq \sqrt{d}$ for all $(\boldsymbol{x}, \boldsymbol{y}), (\boldsymbol{x}', \boldsymbol{y}') \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})$. Hence, if $\gamma > \sqrt{d}$, then finding γ-approximate fixed points of M is trivial and the same is true for fiding α-approximate fixed points of F_{GDA}, since $\gamma = \alpha^2 / 4(G + 2\sqrt{d})$ which implies that, if $\gamma > \sqrt{d}$, then $\alpha > \sqrt{d}$. Thus, we may assume that $\gamma \leq \sqrt{d}$.

Next, to simplify notation we define $(\boldsymbol{x}_\Delta, \boldsymbol{y}_\Delta) = (x^\star - \nabla_x f(\boldsymbol{x}^\star, \boldsymbol{y}^\star), \boldsymbol{y}^\star + \nabla_y f(\boldsymbol{x}^\star, \boldsymbol{y}^\star))$ and $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) = \operatorname{argmin}_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{P}(\boldsymbol{A}, \boldsymbol{b})} \|(\boldsymbol{x}_\Delta, \boldsymbol{y}_\Delta) - (\boldsymbol{x}, \boldsymbol{y})\|_2$. Given that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is a γ-approximate fixed point of M, we have that

$$\|(\boldsymbol{x}^\star, \boldsymbol{y}^\star) - (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})\|_2 < \gamma. \tag{B.6}$$

Using Theorem 1.5.5 (b) of [FP07], we get that

$$\langle (x_\Delta, y_\Delta) - (\hat{x}, \hat{y}), (x, y) - (\hat{x}, \hat{y}) \rangle \leq 0 \text{ for all } (x, y) \in \mathcal{P}(A, b). \tag{B.7}$$

Next we show the following:

Claim B.2. *For all* $(x, y) \in \mathcal{P}(A, b)$, $\langle (x_\Delta, y_\Delta) - (x^\star, y^\star), (x, y) - (x^\star, y^\star) \rangle < (G + 2\sqrt{d}) \cdot \gamma$.

Proof. We have that:

$$\begin{aligned}
\langle (x_\Delta, y_\Delta) - (x^\star, y^\star), (x, y) - (x^\star, y^\star) \rangle \quad &= \quad \langle (x_\Delta, y_\Delta) - (\hat{x}, \hat{y}), (x, y) - (x^\star, y^\star) \rangle \\
&+ \quad \langle (\hat{x}, \hat{y}) - (x^\star, y^\star), (x, y) - (x^\star, y^\star) \rangle \\
&= \quad \langle (x_\Delta, y_\Delta) - (\hat{x}, \hat{y}), (x, y) - (\hat{x}, \hat{y}) \rangle \\
&+ \quad \langle (x_\Delta, y_\Delta) - (\hat{x}, \hat{y}), (\hat{x}, \hat{y}) - (x^\star, y^\star) \rangle \\
&+ \quad \langle (\hat{x}, \hat{y}) - (x^\star, y^\star), (x, y) - (x^\star, y^\star) \rangle \\
&< \quad \|(x_\Delta, y_\Delta) - (\hat{x}, \hat{y})\|_2 \gamma + \gamma \cdot \sqrt{d} \\
&\leq \quad \|(x_\Delta, y_\Delta) - (x^\star, y^\star)\|_2 \gamma + \gamma^2 + \gamma \cdot \sqrt{d} \\
&= \quad \|\nabla f(x^\star, y^\star)\|_2 \gamma + \gamma^2 + \gamma \cdot \sqrt{d} \\
&\leq \quad (G + 2\sqrt{d}) \cdot \gamma,
\end{aligned}$$

where (1) for the first inequality we use (B.6), (B.7), the Cauchy-Schwarz inequality, and the fact that the ℓ_2 diameter of $\mathcal{P}(A, b)$ is at most $\sqrt{d}$; (2) for the second inquality we use the triangle inequality and (B.6); (3) for the equality that follows we use the definition of (x_Δ, y_Δ); and (4) for the last inequality we use that G, the Lipschitzness of f, bounds the magnitude of its gradient, and that $\gamma \leq \sqrt{d}$. □

Now let $x' = \text{argmin}_{x \in K(y^\star)} \|x - x_\Delta\|_2$ where $K(y^\star) = \{x \mid (x, y^\star) \in \mathcal{P}(A, b))\}$. Using Theorem 1.5.5 (b) of [FP07] for x' we get that $\langle x_\Delta - x', x^\star - x' \rangle \leq 0$. Using Claim B.2 for vector $(x', y^\star) \in \mathcal{P}(A, b)$ we get that $\langle x^\star - x_\Delta, x^\star - x' \rangle < (G + 2\sqrt{d})\gamma$. Adding the last two inequalities and using the fact that $\gamma = \alpha^2/4(G + 2\sqrt{d})$ we get the following

$$\left\| x^\star - \Pi_{K(y^\star)}(x^\star - \nabla_x f(x^\star, y^\star)) \right\|_2 < \sqrt{(G + 2\sqrt{d}) \cdot \gamma} = \alpha/2.$$

Using the exact same reasoning we can also prove that

$$\left\| y^\star - \Pi_{K(x^\star)}(y^\star - \nabla_y f(x^\star, y^\star)) \right\|_2 < \alpha/2$$

where $K(x^\star) = \{y \mid (x^\star, y) \in \mathcal{P}(A, b))\}$. Combining the last two inequalities we get that $(x^\star, y^\star)$ is an α-approximate fixed point of F_{GDA}.

C Missing Proofs from Section 8

In this section we present the missing proofs from Section 8 and more precisely in the following sections we prove the Lemmas 8.10, 8.11, and 8.12. For the rest of the proofs in this section we define $L(c)$ to be the cubelet which has the down-left corner equal to c, formaly

$$L(c) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$$

and we also define $L_c(c)$ to be the set of corners of the cubelet $L(c)$, or more formally

$$L_c(c) = \{c_1, c_1 + 1\} \times \cdots \times \{c_d, c_d + 1\}.$$

C.1 Proof of Lemma 8.10

We start with a lemma about the differentiability properties of the functions Q_v^c which we defined in Definition 8.7.

Lemma C.1. *Let $\boldsymbol{x} \in [0,1]^d$ lying in cublet $R(\boldsymbol{x}) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$, where $\boldsymbol{c} \in ([N]-1)^d$. Then for any vertex $\boldsymbol{v} \in R_c(\boldsymbol{x})$, the function $Q_v^c(\boldsymbol{x})$ is continuous and twice differentiable. Moreover if $Q_v^c(\boldsymbol{x}) = 0$ then also $\frac{dQ_v^c(\boldsymbol{x})}{dx_i} = 0$ and $\frac{d^2Q_v^c(\boldsymbol{x})}{dx_i\, dx_j} = 0$.*

Proof. **1st order differentiability:** We remind from the Definition 8.7 that if we let $\boldsymbol{s}^c = (s_1, \ldots, s_d)$ be the source vertex of $R(\boldsymbol{x})$ and $\boldsymbol{p}_{\boldsymbol{x}}^c = (p_1, \ldots, p_d)$ be the canonical representation of $\boldsymbol{x}$. Then for each vertex $\boldsymbol{v} \in R_c(\boldsymbol{x})$ we define the following partition of the set of coordinates $[d]$,

$$A_v^c = \{j : \ |v_j - s_j| = 0\} \text{ and } B_v^c = \{j : \ |v_j - s_j| = 1\}.$$

Now in case $B_v^c = \varnothing$, which corresponds to $\boldsymbol{v}$ being the source node $\boldsymbol{s}^c$ then $Q_v^c(\boldsymbol{x}) = \prod_{j=1}^d S_\infty(1 - S(p_j))$ which is clearly differentiable as product of compositions of differentiable functions. The exact same holds for $A_v^c = \varnothing$ which corresponds to $\boldsymbol{v}$ being the target vertex $\boldsymbol{t}^c$ of the cubelet $R(\boldsymbol{x})$. We thus focus on the case where $A_v^c, B_v^c \neq \varnothing$. To simplify notation we denote $Q_v^c(\boldsymbol{x})$ by $Q(\boldsymbol{x})$, A_v^c by A and B_v^c by B for the rest of this proof. We prove that in case $i \in B$ then $\frac{\partial Q(\boldsymbol{x})}{\partial x_i}$ always exits. The case $i \in A$ follows then symmetrically. We have the following cases

- Let $j \in A$ and $\ell \in B \setminus \{i\}$ such that $p_j \geq p_\ell$. By Definition 8.7, if ε is sufficiently small then $Q(x_i - \varepsilon, \boldsymbol{x}_{-i}) = Q(x_i + \varepsilon, \boldsymbol{x}_{-i}) = Q(x_i, \boldsymbol{x}_{-i}) = 0$. Thus $\frac{\partial Q(\boldsymbol{x})}{\partial x_i}$ exists and equals 0.
- Let $p_\ell > p_j$ for all $\ell \in B \setminus \{i\}$ and $j \in A$. In this case we have the following subcases.
  - $p_i > p_j$ for all $j \in A$: Then $\frac{\partial Q(\boldsymbol{x})}{\partial x_i}$ exists since both $S_\infty(\cdot)$ and $S(\cdot)$ are differentiable.
  - $p_i < p_j$ for some $j \in A$: By Definition 8.7, if ε is sufficiently small then $Q(x_i - \varepsilon, x_{-i}) = Q(x_i + \varepsilon, \boldsymbol{x}_{-i}) = Q(x_i, \boldsymbol{x}_{-i}) = 0$. Thus $\frac{\partial Q(\boldsymbol{x})}{\partial x_i}$ exists and equals 0.
  - $p_i = p_j$ for some $j \in A$ and $p_i \geq p_{j'}$ for all $j' \in A \setminus \{j\}$: By Definition 8.7, if ε is sufficiently small then $Q(x_i - \varepsilon, \boldsymbol{x}_{-i}) = 0$ and also $Q(x_i, \boldsymbol{x}_{-i}) = 0$, thus

    $$\lim_{\varepsilon \to 0^+} \frac{Q(x_i, \boldsymbol{x}_{-i}) - Q(x_i - \varepsilon, \boldsymbol{x}_{-i})}{\varepsilon} = 0.$$

    At the same time

    $$\lim_{\varepsilon \to 0^+} \frac{Q(x_i + \varepsilon, x_{-i}) - Q(x_i, x_{-i})}{\varepsilon} = 0$$

    since both $S_\infty(\cdot)$ and $S(\cdot)$ are differentiable functions, $S_\infty(S(p_i) - S(p_j)) = S_\infty(0) = 0$, and $S'_\infty(S(p_i) - S(p_j)) = S'_\infty(0) = 0$.

2nd order differentiability: Let $Q'(\boldsymbol{x})$ be equal to $\frac{\partial Q(\boldsymbol{x})}{\partial x_k}$ for convenience. As in the previous analysis in case $A_v^c = \varnothing$ or $B_v^c = \varnothing$ then $Q'(x)$ is differentiable with respect to x_i since $S(\cdot), S_\infty(\cdot)$ are twice differentiable. Thus we again focus in the case where $A, B \neq \varnothing$. Notice that by the previous analysis $Q'(\boldsymbol{x}) = 0$ if there exists $\ell \in B$ and $j \in A$ such that $p_\ell \geq p_j$. Without loss of generality we assume that $i \in B$ and we prove that $\frac{\partial Q'(\boldsymbol{x})}{\partial x_i} \triangleq \frac{\partial^2 Q(\boldsymbol{x})}{\partial x_i \partial x_k}$ always exists.

- Let $j \in A$ and $\ell \in B \setminus \{i\}$ such that $p_j \geq p_\ell$. By Definition 8.7, $Q'(x_i - \varepsilon, \boldsymbol{x}_{-i}) = Q'(x_i + \varepsilon, \boldsymbol{x}_{-i}) = Q'(x_i, \boldsymbol{x}_{-i}) = 0$. Thus $\frac{\partial Q'(\boldsymbol{x})}{\partial x_i} \triangleq \frac{\partial^2 Q'(\boldsymbol{x})}{\partial x_i \partial x_k}$ exists and equals 0.
- Let $p_\ell > p_j$ for all $\ell \in B \setminus \{i\}$ and $j \in A$.
  - $p_i > p_j$ for all $j \in A$: Then $\frac{\partial Q'(\boldsymbol{x})}{\partial x_i} \triangleq \frac{\partial^2 Q(\boldsymbol{x})}{\partial x_i \partial x_k}$ exists since both $S_\infty(\cdot)$ and $S(\cdot)$ are twice differentiable.
  - $p_i < p_j$ for some $j \in A$. By Definition 8.7, $Q'(x_i - \varepsilon, \boldsymbol{x}_{-i}) = Q'(x_i + \varepsilon, \boldsymbol{x}_{-i}) = Q'(x_i, \boldsymbol{x}_{-i}) = 0$. Thus $\frac{\partial Q'(\boldsymbol{x})}{\partial x_i} \triangleq \frac{\partial^2 Q(\boldsymbol{x})}{\partial x_i \partial x_k}$ exists and equals 0.
  - $p_i = p_j$ for some $j \in A$ and $p_i > p_{j'}$ for all $j' \in A \setminus \{j\}$. By Definition 8.7, if ε is sufficiently small then $Q'(x_i - \varepsilon, \boldsymbol{x}_{-i}) = 0$ and thus

$$\lim_{\varepsilon \to 0^+} \frac{Q'(x_i, \boldsymbol{x}_{-i}) - Q'(x_i - \varepsilon, \boldsymbol{x}_{-i})}{\varepsilon} = 0.$$

    At the same time $\lim_{\varepsilon \to 0^+} \frac{Q'(x_i + \varepsilon, \boldsymbol{x}_{-i}) - Q'(x_i, \boldsymbol{x}_{-i})}{\varepsilon}$ exists since both $S_\infty(\cdot)$ and $S(\cdot)$ are twice differentiable. Moreover equals 0 since $S_\infty(S(p_i) - S(p_j)) = S_\infty(0) = 0$ and $S'_\infty(S(p_i) - S(p_j)) = S'_\infty(0) = S''_\infty(0) = S(0) = 0$.

In every step of the above proof where we use properties of S_∞ and S we use Lemma 8.3. □

So far we have established the fact that the functions $Q^c_v(\boldsymbol{x})$ are twice differentiable when $\boldsymbol{x}$ moves within the same cubelet. Next we will show that when $\boldsymbol{x}$ moves from one cubelet to another then the corresponding Q^c_v functions changes value smoothly.

Lemma C.2. *Let $\boldsymbol{x} \in [0,1]^d$ such that there exists a coordinate $i \in [d]$ with the property $R(x_i + \varepsilon, \boldsymbol{x}_{-i}) = \left[\frac{c_1}{N-1}, \frac{c_1+1}{N-1}\right] \times \cdots \times \left[\frac{c_d}{N-1}, \frac{c_d+1}{N-1}\right]$ and $R(x_i - \varepsilon, \boldsymbol{x}_{-i}) = \left[\frac{c'_1}{N-1}, \frac{c'_1+1}{N-1}\right] \times \cdots \times \left[\frac{c'_d}{N-1}, \frac{c'_d+1}{N-1}\right]$, with $\boldsymbol{c}, \boldsymbol{c}' \in ([N-1]-1)^d$ and ε sufficiently small, i.e. $\boldsymbol{x}$ lies in the boundary of two cubelets. Then the following statements hold.*

1. *For all vertices $\boldsymbol{v} \in R_c(x_i + \varepsilon, \boldsymbol{x}_{-i}) \cap R_c(x_i - \varepsilon, \boldsymbol{x}_{-i})$, it holds that*
   (a) $Q^c_v(\boldsymbol{x}) = Q^{c'}_v(\boldsymbol{x})$,
   (b) $\frac{\partial Q^c_v(\boldsymbol{x})}{\partial x_j} = \frac{\partial Q^{c'}_v(\boldsymbol{x})}{\partial x_i}$ *for all $i \in [d]$, and*
   (c) $\frac{\partial^2 Q^c_v(\boldsymbol{x})}{\partial x_i \, \partial x_j} = \frac{\partial Q^{c'}_v(\boldsymbol{x})}{\partial x_i \, \partial x_j}$ *for all $i, j \in [d]$.*
2. *For all vertices $\boldsymbol{v} \in R_c(x_i + \varepsilon, \boldsymbol{x}_{-i}) \setminus R_c(x_i - \varepsilon, \boldsymbol{x}_{-i})$, it holds that $Q^c_v(\boldsymbol{x}) = \frac{\partial Q^c_v(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 Q^c_v(\boldsymbol{x})}{\partial x_i \, \partial x_j} = 0$.*
3. *for all vertices $\boldsymbol{v} \in R_c(x_i - \varepsilon, \boldsymbol{x}_{-i}) \setminus R_c(x_i + \varepsilon, \boldsymbol{x}_{-i})$, it holds that $Q^{c'}_v(\boldsymbol{x}) = \frac{\partial Q^{c'}_v(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 Q^{c'}_v(\boldsymbol{x})}{\partial x_i \, \partial x_j} = 0$.*

Lemma C.2 is crucial since it establishes that $\mathsf{P}_v(\boldsymbol{x})$ is a continuous and twice differentiable even when $\boldsymbol{x}$ moves from one cubelet to another. Since the proof of Lemma C.2 is very long and contains the proof of some sublemmas, we postpone it for the end of this section in Section C.1.1. We now proceed with the proof of Lemma 8.10.

Proof of Lemma 8.10. We first prove that $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})$ is a continuous function. Let $\boldsymbol{x} \in [0,1]^d$ lying on the boundary of the following cubelets

$$\left[\frac{c_1^{(1)}}{N-1}, \frac{c_1^{(1)}+1}{N-1}\right] \times \cdots \times \left[\frac{c_d^{(1)}}{N-1}, \frac{c_d^{(1)}+1}{N-1}\right]$$
$$\cdots$$
$$\left[\frac{c_1^{(i)}}{N-1}, \frac{c_1^{(i)}+1}{N-1}\right] \times \cdots \times \left[\frac{c_d^{(i)}}{N-1}, \frac{c_d^{(i)}+1}{N-1}\right]$$
$$\cdots$$
$$\left[\frac{c_1^{(m)}}{N-1}, \frac{c_1^{(m)}+1}{N-1}\right] \times \cdots \times \left[\frac{c_d^{(m)}}{N-1}, \frac{c_d^{(m)}+1}{N-1}\right].$$

where $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(m)} \in ([N-1]-1)^d$. This means that for every $i \in [m]$ there exists a coordinate $j_i \in [d]$ and a value $\eta_i \in \mathbb{R}$ with sufficiently small absolute value such that

$$R(x_{j_i} + \eta_i, \boldsymbol{x}_{-j_i}) = \left[\frac{c_1^{(i)}}{N-1}, \frac{c_1^{(i)}+1}{N-1}\right] \times \cdots \times \left[\frac{c_d^{(i)}}{N-1}, \frac{c_d^{(i)}+1}{N-1}\right].$$

We then consider the following cases.

- $\boldsymbol{v} \notin \cup_{i=1}^m R_c(x_{j_i} + \eta_i, \boldsymbol{x}_{-j_i})$. By Definition 8.9, in all the m aforementioned cubelets, the coefficient $\mathsf{P}_{\boldsymbol{v}}$ takes value 0 and hence it is continuous in this part of the space.
- $\boldsymbol{v} \in \cap_{j \in U} R_c(x_{j_i} + \eta_i, \boldsymbol{x}_{-j_i})$ and $\boldsymbol{v} \notin \cup_{i \in \overline{U}} R_c(x_{j_i} + \eta_i, \boldsymbol{x}_{-j_i})$, for some $U \subseteq [m]$ with $\overline{U} = [m] \setminus U$. In this case $\mathsf{P}_{\boldsymbol{v}}(x_{j_i} + \eta_i, \boldsymbol{x}_{j_i})$ was computed according to a cubelet with $\boldsymbol{v} \in R_c(x_{j_i} + \eta_i, \boldsymbol{x}_{-j_i})$. Then Lemma C.2 implies that $Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i)}}(\boldsymbol{x}) = 0$ since $\boldsymbol{v} \in R_c(x_{j_i} + \eta_i, \boldsymbol{x}_{-j_i}) \setminus R_c(x_{j_{i'}} + \eta_{i'}, \boldsymbol{x}_{-j_{i'}})$ where $i' \in [m]$ and $i \neq i'$. Therefore we conclude that $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x}) = 0$ and

$$\lim_{\eta_i \to 0} \mathsf{P}_{\boldsymbol{v}}(x_{j_i} + \eta_i, \boldsymbol{x}_{-i}) = 0.$$

- $\boldsymbol{v} \in \cap_{i=1}^m R_c(x_{j_i} + \eta_i, \boldsymbol{x}_{-j_i})$. By Lemma C.2 for all $i \in [m]$ it holds that

$$\frac{Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i)}}(\boldsymbol{x})}{\sum_{\boldsymbol{v} \in R_c(x_{j_i}+\eta_i, \boldsymbol{x}_{-j_i})} Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i)}}(\boldsymbol{x})} = \frac{Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i)}}(\boldsymbol{x})}{\sum_{\boldsymbol{v} \in \cap_{i=1}^m R_c(x_{j_i}+\eta_i, \boldsymbol{x}_{-j_i})} Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i)}}(\boldsymbol{x})}$$
$$= \frac{Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i')}}(\boldsymbol{x})}{\sum_{\boldsymbol{v} \in \cap_{i=1}^m R_c(x_{j_i}+\eta_i, \boldsymbol{x}_{-j_i})} Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i')}}(\boldsymbol{x})} = \frac{Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i')}}(\boldsymbol{x})}{\sum_{\boldsymbol{v} \in R_c(x_{j_i}+\eta_i, \boldsymbol{x}_{-j_i})} Q_{\boldsymbol{v}}^{\boldsymbol{c}^{(i')}}(\boldsymbol{x})}$$

  which again implies the continuity of $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})$ at $\boldsymbol{x}$.

Next we prove that $\mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})$ is differentiable for all $\boldsymbol{v} \in ([N]-1)^d$. Fix some $i \in [d]$ we will prove that $\frac{\partial \mathrm{P}(\boldsymbol{x})}{\partial x_i}$ always exists. Let C^+ be the set of down-left corners of the cubelets in which $\lim_{\varepsilon \to 0^+}(x_i + \varepsilon, \boldsymbol{x}_{-i})$ belongs to and C^- be the set of down-left corners of the cubelets in which $\lim_{\varepsilon \to 0^+}(x_i - \varepsilon, \boldsymbol{x}_{-i})$ belongs to. It easy to see that C^+ and C^- are non-empty and fixed for $\varepsilon > 0$ and sufficiently small.

To prove that $\frac{\partial \mathsf{P}_{\boldsymbol{v}}(\boldsymbol{x})}{\partial x_i}$ always exists, we consider the following 3 mutually exclusive cases.

- $\underline{v \in L_c(\boldsymbol{c}^{(1)}) \text{ for } \boldsymbol{c}^{(1)} \in C^+ \text{ and } v \in L_c(\boldsymbol{c}^{(2)}) \text{ for } \boldsymbol{c}^{(2)} \in C^-}$. Since the coefficient $\mathsf{P}_v(\boldsymbol{x})$ is a continuous function, we have that

  - $\lim_{\varepsilon\to 0^+} \frac{\mathsf{P}_v(x_i+\varepsilon,\boldsymbol{x}_{-i})-\mathsf{P}_v(x_i,\boldsymbol{x}_{-i})}{\varepsilon} = \frac{\frac{\partial Q_v^{\boldsymbol{c}^{(1)}}(\boldsymbol{x})}{\partial x_i}\sum_{v'\in L_c(\boldsymbol{c}^{(1)})} Q_{v'}^{\boldsymbol{c}^{(1)}}(\boldsymbol{x}) - Q_v^{\boldsymbol{c}^{(1)}}(\boldsymbol{x})\sum_{v'\in L_c(\boldsymbol{c}^{(1)})}\frac{\partial Q_{v'}^{\boldsymbol{c}^{(1)}}(\boldsymbol{x})}{\partial x_i}}{\left(\sum_{v'\in L_c(\boldsymbol{c}^{(1)})} Q_{v'}^{\boldsymbol{c}^{(1)}}(\boldsymbol{x})\right)^2}$
  - $\lim_{\varepsilon\to 0^+} \frac{\mathsf{P}_v(x_i,\boldsymbol{x}_{-i})-\mathsf{P}_v(x_i-\varepsilon,\boldsymbol{x}_{-i})}{\varepsilon} = \frac{\frac{\partial Q_v^{\boldsymbol{c}^{(2)}}(\boldsymbol{x})}{\partial x_i}\sum_{v'\in L_c(\boldsymbol{c}^{(2)})} Q_{v'}^{\boldsymbol{c}^{(2)}}(\boldsymbol{x}) - Q_v^{\boldsymbol{c}^{(2)}}(\boldsymbol{x})\sum_{v'\in L_c(\boldsymbol{c}^{(2)})}\frac{\partial Q_{v'}^{\boldsymbol{c}^{(2)}}(\boldsymbol{x})}{\partial x_i}}{\left(\sum_{v'\in L_c(\boldsymbol{c}^{(2)})} Q_{v'}^{\boldsymbol{c}^{(2)}}(\boldsymbol{x})\right)^2}$

  Both of the above limits exists due to the fact that $Q_v^{\boldsymbol{c}}(\boldsymbol{x})$ is differentiable (Lemma C.1). Moreover, since $v \in L_c(\boldsymbol{c}^{(1)}) \cap L_c(\boldsymbol{c}^{(2)})$, Case 1 of Lemma C.2 implies that the two limits above have exactly the same value and hence P_v is differentiable at $\boldsymbol{x}$.

- $\underline{v \notin L_c(\boldsymbol{c}^{(1)}) \text{ for all } \boldsymbol{c}^{(1)} \in C^+}$. In the case where $v \notin L_c(\boldsymbol{c})$ for all the down-left corners $\boldsymbol{c}$ of the cubelets at which $\boldsymbol{x}$ lies, then by Definition 8.9 $\mathsf{P}_v(x_i,\boldsymbol{x}_{-i}) = \mathsf{P}_v(x_i+\varepsilon,\boldsymbol{x}_{-i}) = \mathsf{P}_v(x_i-\varepsilon,\boldsymbol{x}_{-i}) = 0$. Thus $\frac{\partial \mathsf{P}_v(\boldsymbol{x})}{\partial x_i}$ exists and equals 0. Therefore we may assume that $v \in L_c(\boldsymbol{c})$ for some down-left corner $\boldsymbol{c}$ of a cubelet at which $\boldsymbol{x}$ lies. Due to the fact that $\mathsf{P}_v(\boldsymbol{x})$ is a continuous function and that $v \notin L_c(\boldsymbol{c}^{(1)})$ for all $\boldsymbol{c}^{(1)} \in C^+$, we get that

  $$\mathsf{P}_v(x_i+\varepsilon,\boldsymbol{x}_{-i}) = 0 \quad \text{and} \quad \mathsf{P}_v(x_i,\boldsymbol{x}_{-i}) = 0.$$

  We also have that $v \in L_c(\boldsymbol{c})/L_c\boldsymbol{c}^{(1)}$ where $\boldsymbol{c}$, $\boldsymbol{c}^{(1)}$ are down-left corners of cubelets at which $\boldsymbol{x}$ lies and $(x_i+\varepsilon,\boldsymbol{x}_{-i})$ lies respectively. Therefore we get by Case 1 of Lemma C.2 that $Q_v^{\boldsymbol{c}}(\boldsymbol{x}) = 0$ implying that $\mathsf{P}_v(x_i,\boldsymbol{x}_{-i}) = 0$. As a result,

  $$\lim_{\varepsilon\to 0^+} \frac{\mathsf{P}_v(x_i+\varepsilon,\boldsymbol{x}_{-i}) - \mathsf{P}_v(x_i,x_{-i})}{\varepsilon} = 0$$

  We now need to argue that $\lim_{\varepsilon\to 0^+} \frac{\mathsf{P}_v(x_i,\boldsymbol{x}_{-i})-\mathsf{P}_v(x_i-\varepsilon,x_{-i})}{\varepsilon}$ exists and equals 0. At first observe that $0 \le x_i - c_i \le \delta$ since $\boldsymbol{x}$ lies in the cubelet with down-left corner $\boldsymbol{c}$. In case $x_i - c_i < \delta$ then $(x_i+\varepsilon,\boldsymbol{x}_{-i})$ lies in $\boldsymbol{c}$ for arbitrarily small ε, meaning that $\boldsymbol{c} \in C^+$. The latter contradicts the fact that $v \notin L_c\boldsymbol{c}^{(1)}$ for all $\boldsymbol{c}^{(1)} \in C^+$. As a result, $x_i - c_i = \delta$ which implies that $\boldsymbol{c} \in C^-$ and hence

  $$\lim_{\varepsilon\to 0^+} \frac{\mathsf{P}_v(x_i,\boldsymbol{x}_{-i}) - \mathsf{P}_v(x_i-\varepsilon,\boldsymbol{x}_{-i})}{\varepsilon} = \frac{\frac{\partial Q_v^{\boldsymbol{c}}(\boldsymbol{x})}{\partial x_i}\sum_{v'\in L_c(\boldsymbol{c})} Q_{v'}^{\boldsymbol{c}}(\boldsymbol{x}) - Q_v^{\boldsymbol{c}}(\boldsymbol{x})\sum_{v'\in L_c(\boldsymbol{c})}\frac{\partial Q_{v'}^{\boldsymbol{c}}(\boldsymbol{x})}{\partial x_i}}{\left(\sum_{v'\in L_c(\boldsymbol{c})} Q_{v'}^{\boldsymbol{c}}(\boldsymbol{x})\right)^2}.$$

  The above limit equals to 0 since $Q_v^{\boldsymbol{c}}(\boldsymbol{x}) = \frac{\partial Q_v^{\boldsymbol{c}}(\boldsymbol{x})}{\partial x_i} = 0$ by applying Lemma C.2 due to the fact that $v \in L_c(\boldsymbol{c}) \setminus L_c(\boldsymbol{c}^{(1)})$.

- $\underline{v \notin L_c(\boldsymbol{c}^{(2)}) \text{ for all } \boldsymbol{c}^{(2)} \in C^-}$. Symmetrically with the previous case.

The second order differentiability of $\mathsf{P}_v(\boldsymbol{x})$ can be established using exactly the same arguments for computing the following limit

$$\lim_{\varepsilon,\varepsilon'\to 0} \frac{\mathsf{P}_v(x_i+\varepsilon,x_j+\varepsilon',\boldsymbol{x}_{-i,j}) - \mathsf{P}_v(\boldsymbol{x})}{\varepsilon^2}.$$

The last thing that we need to show to prove Lemma 8.10 is that the set $R_+(\boldsymbol{x})$ has cardinality at most $d+1$ and that it can be computed in poly(d) time. Let $p_{\boldsymbol{x}}^{\boldsymbol{c}} \in [0,1]^d$ be the canonical representation of $\boldsymbol{x}$ with the respect to a cubelet $L(\boldsymbol{c})$ in which $\boldsymbol{x}$ belongs to. We define the source vertex $\boldsymbol{s^c} = (s_1, \ldots, s_d)$ and the target vertex $\boldsymbol{t^c} = (t_1, \ldots, t_d)$ of $L(\boldsymbol{c})$. Once this is done the vertices in $R_+(\boldsymbol{v})$ are exactly the vertices of $L_c(\boldsymbol{c})$ for which it holds that

$$p_\ell > p_j \qquad \text{for all } \ell \in A_{\boldsymbol{v}}^{\boldsymbol{c}}, j \in B_{\boldsymbol{v}}^{\boldsymbol{c}}$$

since for all the others $\boldsymbol{v} \in ([N]-1)^d$ it holds that $Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x}) = 0$, $\nabla Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x}) = 0$, and $\nabla^2 Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x}) = 0$. These vertices $\boldsymbol{v} \in R_+(\boldsymbol{x})$ can be computed in polynomial time as follows: **i)** the coordinates $p_1, \ldots, p_d$ are sorted in increasing order, and **ii)** for each $m = 0, \ldots, d$ compute the vertex $\boldsymbol{v}^{(m)} \in L_c(\boldsymbol{c})$,

$$\boldsymbol{v}_j^m = \begin{cases} s_j & \text{if coordinate } j \text{ belongs in the first } m \text{ coordinates wrt the order of } \boldsymbol{p_x^c} \\ t_j & \text{if coordinate } j \text{ belongs in the last } d-m \text{ coordinates wrt the order of } \boldsymbol{p_x^c} \end{cases}$$

By Definition 8.7 it immediately follows that $R_+(\boldsymbol{x}) \subseteq \{\boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(m)}\}$ from which we get that $|R_+(\boldsymbol{x})| \leq d+1$ and also they can be computed in poly(d) time. □

To finish the proof of Lemma 8.10 we only need the proof of Lemma C.2 which we present in the following section.

C.1.1 Proof of Lemma C.2

Lemma C.3. *Let a point $\boldsymbol{x} \in [0,1]^d$ lying in the boundary of the cubelets with down-left corners $\boldsymbol{c} = (c_1, \ldots, c_{m-1}, c_m, c_{m+1}, \ldots, c_d)$ and $\boldsymbol{c'} = (c_1, \ldots, c_{m-1}, c_m + 1, c_{m+1}, \ldots, c_d)$. Then the canonical representation of $\boldsymbol{x}$ in the cubelet $L(\boldsymbol{c})$ is the same with the the canonical representation of $\boldsymbol{x}$ in the cubelet $L(\boldsymbol{c'})$. More precisely, $\boldsymbol{p_x^c} = \boldsymbol{p_x^{c'}}$.*

Proof. Let c_m be even. By the definition of the canonical representation in Definition 8.6, the source and target of the cubelets $L(\boldsymbol{c})$ and $L(\boldsymbol{c'})$ are respectively,

- $\boldsymbol{s^c} = (s_1, \ldots, s_{m-1}, c_m, s_{m+1}, \ldots, s_d)$,
- $\boldsymbol{t^c} = (t_1, \ldots, s_{m-1}, c_m + 1, t_{m+1}, \ldots, t_d)$,
- $\boldsymbol{s^{c'}} = (s_1, \ldots, s_{m-1}, c_m + 2, s_{m+1}, \ldots, s_d)$,
- $\boldsymbol{t^{c'}} = (t_1, \ldots, t_{m-1}, c_m + 1, t_{m+1}, \ldots, t_d)$.

Hence we get that $p_j = p'_j$ for $j \neq m$. Since $\boldsymbol{x}$ belongs to the boundary of both cublets $L(\boldsymbol{c})$ and $L(\boldsymbol{c'})$ we get that $x_m = c_m + 1$ which implies that $p_m = p'_m = 1$. In case c_m is odd we get that $\boldsymbol{p_x^c} = \boldsymbol{p_x^{c'}}$ but with $p_m = p'_m = 0$. □

Lemma C.4. *Let $\boldsymbol{x} \in [0,1]^d$ lying at the intersection of the cubelets $L(\boldsymbol{c})$, $L(\boldsymbol{c'})$ with down-left corners $\boldsymbol{c} = (c_1, \ldots, c_{m-1}, c_m, c_{m+1}, \ldots, c_d)$, and $\boldsymbol{c'} = (c_1, \ldots, c_{m-1}, c_m + 1, c_{m+1}, \ldots, c_d)$. Then the following statements are true.*

1. *For all vertices $\boldsymbol{v} \in L_c(\boldsymbol{c}) \cap L_c(\boldsymbol{c'})$ it holds that*
   (a) $Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x}) = Q_{\boldsymbol{v}}^{\boldsymbol{c'}}(\boldsymbol{x})$,

(b) $\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i} = \frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial x_i}$,

(c) $\frac{\partial^2 Q_v^c(\boldsymbol{x})}{\partial x_i\, \partial x_j} = \frac{\partial^2 Q_v^{c'}(\boldsymbol{x})}{\partial x_i\, \partial x_j}$.

2. *For all vertices* $\boldsymbol{v} \in L_c(\boldsymbol{c}) \setminus L_c(\boldsymbol{c}')$ *it holds that* $Q_v^c(\boldsymbol{x}) = \frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 Q_v^c(\boldsymbol{x})}{\partial x_i\, \partial x_j} = 0$.

3. *For all vertices* $\boldsymbol{v} \in L_c(\boldsymbol{c}')/L_c(\boldsymbol{c})$ *it holds that* $Q_v^{c'}(\boldsymbol{x}) = \frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 Q_v^{c'}(\boldsymbol{x})}{\partial x_i\, \partial x_j} = 0$.

Proof. 1. Let $\boldsymbol{v} \in L_c(\boldsymbol{c}) \cap L_c(\boldsymbol{c}')$ then we have that

(a) $Q_v^c(\boldsymbol{x}) = Q_v^{c'}(\boldsymbol{x})$. By Lemma C.3 we get that the canonical representation $\boldsymbol{p}_{\boldsymbol{x}}^{\boldsymbol{c}} = \boldsymbol{p}_{\boldsymbol{x}}^{\boldsymbol{c}'}$. Since $Q_v^c(\boldsymbol{x})$ is a function of the canonical representation $\boldsymbol{p}_{\boldsymbol{x}}^{\boldsymbol{c}}$ (see Definition 8.9), it holds that $Q_v^c(\boldsymbol{x}) = Q_v^{c'}(\boldsymbol{x})$ for all vertices $\boldsymbol{v} \in L_c(\boldsymbol{c}) \cap L_c(\boldsymbol{c}')$.

(b) $\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i} = \frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial x_i}$. For $i \neq m$, we get that $\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i} = \frac{1}{t_i - s_i}\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i} = \frac{1}{t_i' - s_i'}\frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial p_i'} = \frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial x_i}$ since $t_i = t_i'$ and $s_i = s_i'$ for all $i \neq m$. The latter argument cannot be applied for the m-th coordinate since $t_m - s_m = -(t_m' - s_m')$. However since $\boldsymbol{x}$ belongs to the boundary of both the cubelets $L(\boldsymbol{c})$ and $L(\boldsymbol{c}')$ it is implied that $p_m = p_m'$ is either 0 or 1, meaning that $\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_m} = \frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial x_m} = 0$ since $S'(0) = S'(1) = 0$ from Lemma 8.3.

(c) $\frac{\partial^2 Q_v^c(\boldsymbol{x})}{\partial x_i\, \partial x_j} = \frac{\partial^2 Q_v^{c'}(\boldsymbol{x})}{\partial x_i\, \partial x_j}$. For $i, j \neq m$, we get that $\frac{\partial^2 Q_v^c(\boldsymbol{x})}{\partial x_i\, \partial x_j} = \frac{1}{t_i - s_i}\frac{1}{t_j - s_j}\frac{\partial^2 Q_v^c(\boldsymbol{x})}{\partial p_i\, \partial p_j} = \frac{1}{t_i' - s_i'}\frac{1}{t_j' - s_j'}\frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial p_i'\, \partial p_j'} = \frac{\partial^2 Q_v^{c'}(\boldsymbol{x})}{\partial x_i\, \partial x_j}$ since $t_i = t_i'$ and $s_i = s_i'$ for all $i \neq m$. As in the previous case, $p_m = p_m'$ equals either 0 or 1. As a result, $\frac{\partial^2 Q_v^c(\boldsymbol{x})}{\partial x_m\, \partial x_j} = \frac{\partial^2 Q_v^{c'}(\boldsymbol{x})}{\partial x_m\, \partial x_j} = 0$ since $S'(0) = S'(1) = S''(0) = S''(1) = 0$ by Lemma 8.3.

2. Since $\boldsymbol{v} \in L_c(\boldsymbol{c}) \setminus L_c(\boldsymbol{c}')$, we get that $v_m = c_m$. In case c_m is even, we get that $s_m = c_m = v_m$ and thus the coordinate the coordinate m belongs in the set $A_{\boldsymbol{v}}^{\boldsymbol{c}}$. Since $\boldsymbol{x}$ coincides with one of the corners in $L_c(\boldsymbol{c}) \setminus L_c(\boldsymbol{c}')$ we get that $p_m = 1$ which combined with the fact that $m \in A_{\boldsymbol{v}}^{\boldsymbol{c}}$ implies that $Q_v^c(\boldsymbol{x}) = 0$ (see Definition 8.7). Then by Lemma C.1, $\frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 Q_v^{c'}(\boldsymbol{x})}{\partial x_i\, \partial x_j} = 0$. In case is odd, we get that $s_m = c_m + 1$. The latter combined with the fact that $v_m = c_m$ implies that the m-th coordinate belongs in $B_{\boldsymbol{v}}^{\boldsymbol{c}}$. Now $p_m = 0$ and by Definition 8.7, $Q_v^c(\boldsymbol{x}) = 0$. Then again by Lemma C.1, $\frac{\partial Q_v^{c'}(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 Q_v^{c'}(\boldsymbol{x})}{\partial x_i\, \partial x_j} = 0$.

3. This case follows with the same reasoning with previous case 2.

□

We are now ready to prove Lemma C.2.

Proof of Lemma C.2. 1. Let $\boldsymbol{v} \in L_c(\boldsymbol{c}) \cap L_c(\boldsymbol{c}')$. There exists a sequence of corners

$$\boldsymbol{c} = \boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(m)} = \boldsymbol{c}'$$

such that $\left\|\boldsymbol{c}^{(j)} - \boldsymbol{c}^{(j+1)}\right\|_1 = 1$ and $\boldsymbol{v} \in L_c(\boldsymbol{c}^j)$ for all $j \in [m]$. By Lemma C.4 we get that,

(a) $Q_v^{\boldsymbol{c}^{(j)}}(\boldsymbol{x}) = Q_v^{\boldsymbol{c}^{(j+1)}}(\boldsymbol{x})$.

(b) $\frac{\partial Q_v^{\boldsymbol{c}^{(j)}}(\boldsymbol{x})}{\partial x_i} = \frac{\partial Q_v^{\boldsymbol{c}^{(j+1)}}(\boldsymbol{x})}{\partial x_i}$.

(c) $\frac{\partial^2 Q_v^{c^{(j)}}(\boldsymbol{x})}{\partial x_i\, \partial x_j} = \frac{\partial Q_v^{c^{(j+1)}}(\boldsymbol{x})}{\partial x_i\, \partial x_j}$.

which implies Case 1 of Lemma C.2.

2. Let $\boldsymbol{v} \in L_c(\boldsymbol{c}) \setminus L_c(\boldsymbol{c}')$. There exists a sequence of corners $\boldsymbol{c} = \boldsymbol{c}^{(1)} \ldots, \boldsymbol{c}^{(i)}$ such that $\left\|\boldsymbol{c}^{(j)} - \boldsymbol{c}^{(j+1)}\right\|_1 = 1$ and $\boldsymbol{v} \notin L_c \boldsymbol{c}^{(i)}$ and $\boldsymbol{v} \in L_c(\boldsymbol{c}^{(j)})$ for all $j < i$. By case 2 of Lemma C.4 we get that $Q_v^{c^{(i-1)}}(\boldsymbol{x}) = \frac{\partial Q_v^{c^{(i-1)}}(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 Q_v^{c^{(i-1)}}(\boldsymbol{x})}{\partial x_i\, \partial x_j} = 0$. Then case 2 of Lemma C.2 follows by case 1 of Lemma C.4.

3. Similarly with case 2.

□

C.2 Proof of Lemma 8.11

We start this section with some fundamental properties of the smooth step function S_∞ that are more fine-grained than the properties we presented in Lemma 8.3.

Lemma C.5. *For $d \geq 10$ there exists a universal constant $c > 0$ such that the following statements hold.*

1. *If $x \geq 1/d$ then $S_\infty(x) \geq c \cdot 2^{-d}$.*
2. *If $x \leq 1/d$ then $S'_\infty(x) \leq c \cdot d^2 \cdot 2^{-d}$.*
3. *If $x \geq 1/d$ then $\frac{S'_\infty(x)}{S_\infty(x)} \leq c \cdot d^2$.*
4. *If $x \leq 1/d$ then $|S''_\infty(x)| \leq c \cdot d^4 \cdot 2^{-d}$.*
5. *If $x \geq 1/d$ then $\frac{|S''_\infty(x)|}{S_\infty(x)} \leq c \cdot d^4$.*

Proof. We compute the derivative of S_∞ and we have that

$$S'_\infty(x) = \ln(2) S_\infty(x) S_\infty(1-x) \left(\frac{1}{x^2} + \frac{1}{(1-x)^2}\right)$$

from which we immediately get $S'_\infty(x) \geq 0$. Then we can compute the second derivative of S_∞ as follows

$$S''_\infty(x) = \ln(2) S_\infty(x) S_\infty(1-x) \cdot$$
$$\cdot \left(\ln(2) \left(S_\infty(1-x) - S_\infty(x)\right) \left(\frac{1}{x^2} + \frac{1}{(1-x)^2}\right)^2 - 2\left(\frac{1}{x^3} - \frac{1}{(1-x)^3}\right)\right).$$

We next want to prove that $S''_\infty(x) \geq 0$ for $x \leq 1/10$. To see this observe that $1 - 2 \cdot S_\infty(x) \geq 1/2$ for $x \leq 1/d$ and therefore

$$S''_\infty(x) \geq \frac{\ln(2)}{x^3} S_\infty(x) S_\infty(1-x) \left(\frac{\ln(2)}{2x} - 2\right)$$

hence for $x \leq 4/\ln(2)$ it holds that $S''_\infty(x) \geq 0$. By similar but more tedious calculations we can conclude that $S'''_\infty(x) \geq 0$ for $x \leq 1/10$. Hence in the interval $x \in [0, 1/10]$ all the functions S_∞, S'_∞, S''_∞ are all increasing functions of x.

Next we show that the function $h(x) = 2^{-1/x} + 2^{-1/(1-x)}$ is upper and lower bounded. First observe that $h(x) \geq \max\{2^{-1/x}, 2^{-1/(1-x)}\}$. Now if we set $t(x) = 2^{-1/x}$ then $t'(x) = \ln(2)t(x)/x^2$ and hence $t(x) \geq t(1/2) = 1/4$ for $x \geq 1/2$. The same way we can prove that $2^{-1/(1-x)} \geq 1/4$ for $x \leq 1/2$. Therefore $h(x) \geq 1/4$ for all $x \in [0,1]$. Also it is not hard to see that $2^{-1/x} \leq 1/2$ and $2^{-1/(1-x)} \leq 1/2$ which implies $h(x) \leq 1$. Hence overall we have that $h(x) \in [1/4, 1]$ for all $x \in [0,1]$. We are now ready to prove the statements.

1. We have shown that $S'_\infty(x) \geq 0$ for all $x \in [0,1]$. Hence S_∞ is an increasing function and therefore $S_\infty(x) \geq S_\infty(1/d)$ for $x \geq 1/d$. Now we have that $S_\infty(1/d) = 2^{-d}/h(1/d) \geq 2^{-d}$.

2. Since $S'_\infty(x)$ is increasing for $x \in [0, 1/10]$, we have that $S'_\infty(x) \leq S'_\infty(1/d)$ for $x \leq 1/d$ and therefore
$$S'_\infty(x) \leq \ln(2) S_\infty(1-1/d) S_\infty(1/d) \left(d^2 + \frac{1}{\left(1 - \frac{1}{d}\right)^2} \right)$$
$$\leq 2\ln(2) \frac{2^{-d}}{h(1/d)} \leq 8\ln(2) 2^{-d}.$$

3. We have that for $x \leq 1/d$
$$\frac{S'_\infty(x)}{S_\infty(x)} = \ln(2) S_\infty(1-x) \left(\frac{1}{x^2} + \frac{1}{(1-x)^2} \right) \leq 2\ln(2) \frac{1}{x^2} \leq 2\ln(2) d^2.$$

4. Follows directly from the statement 1., the fact that $S''_\infty(x)$ is increasing for $x \in [0, 1/10]$ and the above expression of S''_∞ this statement follows.

5. This statement follows using the same reasoning with statement 3.

□

In this section we establish the bounds on the gradient and the hessian of $\mathrm{P}_v(\boldsymbol{x})$. These bounds are formally stated in Lemma 8.11 the proof of which is the main goal of the section.

Lemma 8.11. *For any vertex $\boldsymbol{v} \in ([N]-1)^d$, it holds that*

1. $\left| \frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial x_i} \right| \leq \Theta(d^{12}/\delta)$,
2. $\left| \frac{\partial^2 \mathrm{P}_v(\boldsymbol{x})}{\partial x_i \, \partial x_j} \right| \leq \Theta(d^{24}/\delta^2)$.

In order to prove Lemma 8.11. We first introduce several technical lemmas.

Lemma C.6. *Let $\boldsymbol{x} \in [0,1]^d$ lying in cublet $L(\boldsymbol{c})$, with $\boldsymbol{c} \in ([N]-1)^d$ and let $\boldsymbol{p}^{\boldsymbol{c}}_{\boldsymbol{x}} = (p_1, \ldots, p_d)$ be the canonical representation of $\boldsymbol{x}$. Then for all vertices $\boldsymbol{v} \in L_c(\boldsymbol{c})$, it holds that*
$$\left| \frac{\partial Q^c_v(\boldsymbol{x})}{\partial p_i} \right| \leq \Theta(d^{11}) \cdot \sum_{v \in V_c} Q^c_v(\boldsymbol{x}).$$

Proof. To simplify notation we use $Q_v(\boldsymbol{x})$ instead of $Q_v^c(\boldsymbol{x})$, A instead of A_v^c and B instead of B_v^c for the rest of the proof. Without loss of generality we assume that for all $j \in A$ and $\ell \in B$, $p_\ell > p_j$ since otherwise $\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i} = 0$ trivially by the Definition 8.7. Let $i \in B$ (symmetrically for $i \in A$) then,

$$\begin{aligned}
&\left|\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right| = \\
&\quad = \prod_{\ell \neq i}\prod_{j \in A} S_\infty(S(p_\ell) - S(p_j)) \cdot \left[\sum_{j \in A} \left|S'_\infty(S(p_i) - S(p_j))\right| \prod_{j' \in A/\{j\}} S_\infty(S(p_i) - S(p_{j'}))\right] S'(p_i) \\
&\quad \leq 6 \sum_{j \in A} \left|S'_\infty(S(p_i) - S(p_j))\right| \cdot \prod_{(j',\ell) \neq (j,i)} S_\infty(S(p_\ell) - S(p_{j'}))
\end{aligned}$$

where the last inequality follows by the fact that $|S'(\cdot)| \leq 6$. Since $|A| \leq d$ the proof of the lemma will be completed if we are able to show that for any $j \in A$, it holds that

$$\left|S'_\infty(S(p_i) - S(p_j))\right| \cdot \prod_{(j',\ell) \neq (j,i)} S_\infty(S(p_\ell) - S(p_{j'})) \leq \Theta(d^{10}) \cdot \sum_{\boldsymbol{v}' \in L_c(\boldsymbol{c})} Q_{\boldsymbol{v}'}(\boldsymbol{x})$$

In case $S(p_i) - S(p_j) \geq 1/d^5$ then by case 3. of Lemma C.5 we get that $\left|S'_\infty(S(p_i) - S(p_j))\right| \leq c \cdot d^{10} \cdot S_\infty(S(p_i) - S(p_j))$, which implies gthe following

$$\begin{aligned}
&\left|S'_\infty(S(p_i) - S(p_j))\right| \cdot \prod_{(j',\ell) \neq (j,i)} S_\infty(S(p_\ell) - S(p_{j'})) \leq \\
&\qquad \leq c \cdot d^{10} \cdot S_\infty(S(p_i) - S(p_j)) \cdot \prod_{(j',\ell) \neq (j,i)} S_\infty(S(p_\ell) - S(p_{j'})) \\
&\qquad = c \cdot d^{10} \cdot Q_{\boldsymbol{v}}(\boldsymbol{x}) \\
&\qquad \leq c \cdot d^{10} \cdot \sum_{\boldsymbol{v}' \in L_c(\boldsymbol{c})} Q_{\boldsymbol{v}'}(\boldsymbol{x})
\end{aligned}$$

Now consider the case where $S(p_i) - S(p_j) \leq 1/d^5$. Using case 2. of Lemma C.5, we have that

$$\left|S'_\infty(S(p_i) - S(p_j))\right| \cdot \prod_{(j',\ell) \neq (j,i)} S_\infty(S(p_\ell) - S(p_{j'})) \leq \left|S'_\infty(S(p_i) - S(p_j))\right| \leq \Theta(d^{10} \cdot 2^{-d^5})$$

Consider the sequence of points in the $[0,1]$ interval $0, p_1, \ldots, p_d, 1$. There always exist two consecutive points with distance greater that $1/(d+1)$. As a result, there exists $\boldsymbol{v}^* \in L_c(\boldsymbol{c})$ such that $p_\ell - p_j \geq 1/(d+1)$ for all $\ell \in B_{\boldsymbol{v}^*}$ and $j \in A_{\boldsymbol{v}^*}$. Then $S(p_\ell) - S(p_j) \geq 1/(d+1)^2$ and by case 1. of Lemma C.5, $S_\infty(S(p_\ell) - S(p_j)) \geq c2^{-(d+1)^2}$. If we also use the fact that $|A_{\boldsymbol{v}^*}| \cdot |B_{\boldsymbol{v}^*}| \leq d^2$, we get that

$$Q_{\boldsymbol{v}^*}(\boldsymbol{x}) \geq (c \cdot 2^{-(d+1)^2})^{d^2} = c^{d^2} 2^{-(d+1)^2 \cdot d^2}.$$

Then it holds that

$$\begin{aligned}
&\frac{1}{Q_{\boldsymbol{v}^*}(\boldsymbol{x})} \cdot \left|S'_\infty(S(p_i) - S(p_j))\right| \cdot \prod_{(j',\ell) \neq (j,i)} S_\infty(S(p_\ell) - S(p_{j'})) \leq \\
&\qquad \leq \Theta\left(d^{10} \cdot \left((1/c) \cdot 2^{-d^3 + (d+1)^2}\right)^{d^2}\right) \leq \Theta(d^{10}).
\end{aligned}$$

Combining the later with the discussion in the rest of the proof the lemma follows. □

Lemma C.7. *For any vertex $\boldsymbol{v} \in ([N]-1)^d$ it holds that $\left|\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial x_i}\right| \leq \Theta\left(d^{12}/\delta\right)$.*

Proof. To simplify notation we use $Q_v(\boldsymbol{x})$ instead of $Q_v^{\boldsymbol{c}}(\boldsymbol{x})$ for the rest of the proof. Without loss of generality we assume that $\boldsymbol{x}$ lies on a cubelet $L(\boldsymbol{c})$ with $\boldsymbol{c} \in ([N]-1)^d$ and $\boldsymbol{v} \in L_c(\boldsymbol{c})$, since otherwise $\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial x_i} = 0$. Let $\boldsymbol{p}_{\boldsymbol{x}}^{\boldsymbol{c}} = (p_1, \ldots, p_d)$ be the canonical representation of $\boldsymbol{x}$ in the cubelet $L(\boldsymbol{c})$. Then it holds that

$$\begin{aligned}
\left|\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial p_i}\right| &= \frac{\left|\frac{\partial Q_v(\boldsymbol{x})}{\partial p_i} \cdot \left[\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x})\right] - Q_v(\boldsymbol{x}) \cdot \left[\sum_{v' \in L_c(\boldsymbol{c})} \frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_i}\right]\right|}{(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}))^2} \\
&\leq \frac{\left|\frac{\partial Q_v(\boldsymbol{x})}{\partial p_i}\right|}{\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x})} + \frac{\sum_{v' \in L_c(\boldsymbol{c})} \left|\frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_i}\right|}{\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x})} \\
&\leq (d+2) \cdot \Theta(d^{11}) = \Theta(d^{12})
\end{aligned}$$

where the last inequality follows by Lemma C.6 and the fact that at most $d+1$ vertices $\boldsymbol{v}$ of $L_c(\boldsymbol{c})$ have non-zero gradient as we have proved in Lemma 8.10. Then the proof of Lemma C.7 follows by the fact that $p_i = \frac{x_i - s_i}{t_i - s_i}$. □

Lemma C.8. *Let $\boldsymbol{c} \in ([N]-1)^d$ and $\boldsymbol{v} \in L_c(\boldsymbol{c})$ then it holds that $\left|\frac{\partial^2 Q_v^{\boldsymbol{c}}(\boldsymbol{x})}{\partial p_i\, \partial p_j}\right| \leq \Theta(d^{22}) \cdot \sum_{\boldsymbol{v} \in R_{\boldsymbol{c}}(\boldsymbol{x})} Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x})$.*

Proof. To simplify the notation we use $CS(p_\ell - p_m)$ to denote $S_\infty(S(p_\ell) - S(p_m))$, $CS'(p_\ell - p_m)$ to denote $|S'_\infty(S(p_\ell) - S(p_m))|$, A to denote $A_v^{\boldsymbol{c}}$ and B to denote $B_v^{\boldsymbol{c}}$ for the rest of the proof. As in Lemma C.7, we assume that $p_\ell > p_m$ for all $\ell \in B$ and $m \in A$ since otherwise $\frac{\partial^2 Q_v(\boldsymbol{x})}{\partial p_i\, \partial p_j} = 0$. We have the following cases for the indices i and j

- If $i, j \in B$ then

$$\begin{aligned}
&\left|\frac{\partial^2 \mathrm{Q}_v(\boldsymbol{x})}{\partial p_i\, \partial p_j}\right| = \\
&= \sum_{m_1, m_2 \in A} CS'(p_i - p_{m_1}) CS'(p_j - p_{m_2}) \cdot \prod_{(m,\ell) \neq \{(m_1,i),(m_2,j)\}} CS(p_\ell - p_m) \cdot S'(p_i) S'(p_j) \\
&\leq 36 \sum_{m_1, m_2 \in A} \underbrace{CS'(p_i - p_{m_1}) CS'(p_j - p_{m_2}) \cdot \prod_{(m,\ell) \neq \{(m_1,i),(m_2,j)\}} CS(p_\ell - p_m)}_{\triangleq U(i,j)}.
\end{aligned}$$

  If additionally it holds that $S(p_i) - S(p_{m_1}) \leq 1/d^5$ or $S(p_j) - S(p_{m_2}) \leq 1/d^5$, then by the case 2. of Lemma C.5, we have that

$$U(i,j) \leq CS'(p_i - p_{m_1}) \cdot CS'(p_j - p_{m_2}) \leq \Theta(d^{10} e^{-d^5}).$$

  The latter follows from the fact that the function $S'_\infty(\cdot)$ is bounded in the $[0,1]$ interval and that $CS(p_\ell - p_m) \leq 1$. With the exact same arguments as in Lemma C.6, we hence get that

$$CS'(p_i - p_{m_1}) CS'(p_j - p_{m_2}) \cdot \Pi_{(m,\ell) \neq \{(m_1,i),(m_2,j)\}} CS(p_\ell - p_m) \leq \Theta(d^{10}) \sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}^{\boldsymbol{c}}(\boldsymbol{x}).$$

  Thus $\left|\frac{\partial^2 Q_v(\boldsymbol{x})}{\partial p_i\, \partial p_j}\right| \leq \Theta(d^{12}) \sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}^{\boldsymbol{c}}(\boldsymbol{x})$.

On the other hand if $S(p_i) - S(p_{m_1}) \geq 1/d^5$ and $S(p_j) - S(p_{m_2}) \geq 1/d^5$ then by case 1. of Lemma C.5, $CS'(p_i - p_{m_1}) \leq c \cdot d^{10} \cdot CS(p_i - p_{m_1})$ and $CS'(p_j - p_{m_2}) \leq c \cdot d^{10} \cdot CS(p_j - p_{m_2})$ and thus $U(i,j) \leq \Theta(d^{20}) \cdot Q^c_v(\boldsymbol{x})$. Overall we get that $\left|\frac{\partial^2 Q_v(\boldsymbol{x})}{\partial p_i \, \partial p_j}\right| \leq \Theta(d^{22}) \cdot \sum_{v' \in R_c(\boldsymbol{x})} Q^c_{v'}(\boldsymbol{x})$.

- If $i \in B$ and $j \in A$ then

$$\begin{aligned}
&\left|\frac{\partial^2 \mathrm{Q}_v(\boldsymbol{x})}{\partial p_i \, \partial p_j}\right| \leq \\
&\leq \sum_{m_1 \in A, \ell_2 \in B} CS'(p_i - p_{m_1}) CS'(p_{\ell_2} - p_j) \cdot \prod_{(m,\ell) \neq \{(i,m_1),(\ell_2,j)\}} CS(p_\ell - p_m) \cdot S'(p_i) S'(p_j) \\
&\quad + \left| CS''(p_i - p_j) \cdot \prod_{(m,\ell) \neq (i,j)} CS(p_\ell - p_m) \cdot S'(p_i) S'(p_j) \right| \\
&\leq \Theta(d^{22}) \sum_{v \in L_c(\boldsymbol{c})} Q^c_v(\boldsymbol{x}) + 36 \underbrace{\left| CS''(p_i - p_j) \cdot \prod_{(m,\ell) \neq (i,j)} CS(p_\ell - p_m) \right|}_{Q''(\boldsymbol{x})}.
\end{aligned}$$

  In case $S(p_i) - S(p_j) \geq 1/d^5$ then by case 4. of Lemma C.5, we get that $\left|CS''(p_i - p_j)\right| \leq cd^{20} \cdot CS(p_i - p_j)$ which implies that $Q'' \leq \Theta(d^{20}) \cdot Q^c_v(\boldsymbol{x})$.

  On the other hand if $S(p_i) - S(p_j) \leq 1/d^5$ then by case 5. of Lemma C.5, we get that $Q'' \leq \left|CS''(p_i - p_j)\right| \leq c \cdot d^{20} e^{-d^5}$. As in the proof of Lemma C.6, there exists a vertex $v^* \in R_c(\boldsymbol{x})$ such that $Q^c_{v^*}(\boldsymbol{x}) \geq c^{d^2} e^{-(d+1)^2 d^2}$ and thus $Q'' \leq \Theta(d^{20}) \sum_{v \in L_c(\boldsymbol{c})} Q^c_v(\boldsymbol{x})$. Overall we get that

$$\left|\frac{\partial^2 \mathrm{Q}_v(\boldsymbol{x})}{\partial p_i \, \partial p_j}\right| \leq \Theta(d^{22}) \sum_{v \in L_c(\boldsymbol{c})} Q^c_v(\boldsymbol{x}).$$

- If $i = j \in B$ then

$$\begin{aligned}
&\left|\frac{\partial^2 \mathrm{Q}_v(\boldsymbol{x})}{\partial^2 p_i}\right| \leq \\
&\leq \sum_{m_1, m_2 \in A} \left| CS'(p_i - p_{m_1}) CS'(p_i - p_{m_2}) \cdot \prod_{(m,\ell) \neq \{(m_1,i),(m_2,i)\}} CS(p_\ell - p_m) \cdot S'(p_i) S'(p_i) \right| \\
&\quad + \sum_{m_1 \in A} \left| CS''(p_i - p_{m_1}) \cdot \prod_{(m,\ell) \neq (m_1,\ell)} CS(p_\ell - p_m) S'(p_i) S'(p_i) \right| \\
&\leq \Theta(d^{22} + d \cdot d^{20}) \cdot \sum_{v \in L_c(\boldsymbol{c})} Q^c_v(\boldsymbol{x}).
\end{aligned}$$

If we combine all the above cases then the Lemma follows. □

Lemma C.9. *For any vertex $v \in ([N]-1)^d$, it holds that $\left|\frac{\partial^2 \mathrm{P}_v(\boldsymbol{x})}{\partial x_i \, \partial x_j}\right| \leq \Theta(d^{24}/\delta^2)$.*

Proof. Without loss of generality we assume that $\boldsymbol{v} \in L_c(\boldsymbol{c})$, where $\boldsymbol{c} \in ([N-1]-1)^d$ such that $\boldsymbol{x} \in L(\boldsymbol{c})$, since otherwise $\frac{\partial^2 \mathrm{P}_v(\boldsymbol{x})}{\partial x_i \, \partial x_j} = 0$.

$$
\begin{aligned}
\frac{\partial^2 \mathrm{P}_v(\boldsymbol{x})}{\partial p_i \, \partial p_j} \quad = \quad & \frac{\partial^2 Q_v(\boldsymbol{x})}{\partial p_i \, \partial p_j} \left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^3 \cdot \frac{1}{\left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^4} \\
+ \quad & \frac{\partial Q_v(\boldsymbol{x})}{\partial p_i} \sum_{v' \in L_c(\boldsymbol{c})} \frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_j} \left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^2 \cdot \frac{1}{\left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^4} \\
- \quad & \frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_j} \sum_{v' \in L_c(\boldsymbol{c})} \frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_i} \left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^2 \cdot \frac{1}{\left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^4} \\
- \quad & Q_v(\boldsymbol{x}) \sum_{v' \in L_c(\boldsymbol{c})} \frac{\partial^2 Q_{v'}(\boldsymbol{x})}{\partial p_i \, \partial p_j} \left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^2 \cdot \frac{1}{\left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^4} \\
- \quad & \frac{\partial Q_v(\boldsymbol{x})}{\partial p_i} \sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \cdot 2 \sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \sum_{v' \in L_c(\boldsymbol{c})} \frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_j} \cdot \frac{1}{\left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^4} \\
+ \quad & Q_v(\boldsymbol{x}) \sum_{v' \in L_c(\boldsymbol{c})} \frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_i} \cdot 2 \sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \sum_{v' \in L_c(\boldsymbol{c})} \frac{\partial Q_{v'}(\boldsymbol{x})}{\partial p_j} \cdot \frac{1}{\left(\sum_{v' \in L_c(\boldsymbol{c})} Q_{v'}(\boldsymbol{x}) \right)^4}
\end{aligned}
$$

Using Lemma C.8 and Lemma C.6 we can bound every term in the above expression and hence we get that $\left| \frac{\partial^2 \mathrm{P}_v(\boldsymbol{x})}{\partial p_i \, \partial p_j} \right| \leq \Theta(d^{24})$. Then the lemma follows from the fact that $\frac{\partial p_i}{\partial x_i} = 1/\delta$. □

Finally using Lemma C.7 and Lemma C.9 we get the proof of Lemma 8.11.

C.3 Proof of Lemma 8.12

Let $0 \leq x_i < 1/(N-1)$ and $\boldsymbol{c} = (c_1, \ldots, c_i, \ldots, c_d)$ denote down-left corner of the cubelet $R(\boldsymbol{x})$ at which $\boldsymbol{x} \in [0,1]^d$ lies, i.e. $\boldsymbol{x} \in L(\boldsymbol{c})$. Since $\boldsymbol{x} \leq 1/(N-1)$, this means that $c_i = 0$. By the definition of *sources and targets* in Definition 8.6, we have that $s_i = 0$ and $t_i = 1/(N-1)$, where s_i, t_i are respectively the i-th coordinate of the source $\boldsymbol{s_c}$ and the target $\boldsymbol{t_c}$ vertex. Let the canonical representation $p_x^{\boldsymbol{c}} = (p_1, \ldots, p_d)$ of $\boldsymbol{x}$ in the cubelet $L(\boldsymbol{c})$. Now partition the coordinates $[d]$ in the following sets

$$A = \{ j \mid p_j \leq p_i \} \quad \text{and} \quad B = \{ j \mid p_i < p_j \}.$$

If $B = \varnothing$ then notice that $\mathrm{P}_{\boldsymbol{s_c}}(\boldsymbol{x}) > 0$, since $p_i < 1$, by the fact that $x_i < 1/(N-1)$. Thus the lemma follows since $s_i = 0$. So we may assume that $B \neq \varnothing$. In this case consider the corner $\boldsymbol{v} = (v_1, \ldots, v_d)$ defined as follows

$$v_j = \begin{cases} s_j & j \in A \\ t_j & j \in B \end{cases}.$$

Observe that $Q_v^{\boldsymbol{c}}(\boldsymbol{x}) > 0$ and thus $\boldsymbol{v} \in R_+(\boldsymbol{x})$. Moreover the coordinate $i \in A$ and therefore it holds that $v_i = s_i = 0$. This proves the first statement of the Lemma.

For the second statement let $1 - 1/(N-1) \leq x_i \leq 1/(N-1)$ and $\boldsymbol{c} = (c_1, \ldots, c_i, \ldots, c_d)$ denote down-left corner of the cubelet $R(\boldsymbol{x})$ at which $\boldsymbol{x} \in [0,1]^d$ lies, i.e. $\boldsymbol{x} \in L(\boldsymbol{c})$. This means that $c_i = \frac{N-2}{N-1}$.

- Let N be odd. In this case by the definition of sources and targets in Definition 8.6, we have that $s_i = 1 - 1/(N-1)$ and $t_i = 1$, where s_i, t_i are respectively the i-th coordinate of the source and target vertex. Let $p^c_{\boldsymbol{x}} = (p_1, \ldots, p_d)$ be the canonical representation of $\boldsymbol{x}$ under in the cubelet $L(\boldsymbol{c})$. Now partition the coordinates $[d]$ as follows,

$$A = \{j \mid p_j < p_i\} \quad \text{and} \quad B = \{j \mid p_i \leq p_j\}$$

If $A = \varnothing$ then notice that for the target vertex $\boldsymbol{t_c}$, $\mathsf{P}_{\boldsymbol{t_c}}(\boldsymbol{x}) > 0$, since $p_i > 0$, by the fact that $x_i > 1 - 1/(N-1)$. Thus the lemma follows since $t_i = 1$. So we may assume that $A \neq \varnothing$. In this case consider the corner $\boldsymbol{v} = (v_1, \ldots, v_d)$ defined as follows,

$$v_j = \begin{cases} s_j & j \in A \\ t_j & j \in B \end{cases}$$

Observe that $Q^c_{\boldsymbol{v}}(\boldsymbol{x}) > 0$ and thus $\boldsymbol{v} \in R_+(\boldsymbol{x})$. Moreover the coordinate $i \in B$ and thus $v_i = t_i = 1$.

- Let N be even. In this case we have that $t_i = 1 - 1/(N-1)$ and $s_i = 1$. Now partition the coordinates $[d]$ as follows,

$$A = \{j \mid p_j \leq p_i\} \quad \text{and} \quad B = \{j \mid p_i < p_j\}$$

If $B = \varnothing$ then notice that for the source vertex $\boldsymbol{s_c}$, $\mathsf{P}_{\boldsymbol{s_c}}(\boldsymbol{x}) > 0$, since $p_i < 1$, by the fact that $x_i > 1 - 1/(N-1)$. Thus the lemma follows since $s_i = 1$. In case $B \neq \varnothing$ consider the corner $\boldsymbol{v} = (v_1, \ldots, v_d)$ defined as follows,

$$v_j = \begin{cases} s_j & j \in A \\ t_j & j \in B \end{cases}$$

Observe that $Q^c_{\boldsymbol{v}}(\boldsymbol{x}) > 0$ and thus $\boldsymbol{v} \in R_+(\boldsymbol{x})$. Moreover the coordinate $i \in A$ and thus $v_i = s_i = 1$.

If we put together the last two cases then this implies the second statement of the lemma.

D Constructing the Turing Machine – Proof of Theorem 7.6

In this section we prove Theorem 7.6 establishing that both the function $f_{C_l}(\boldsymbol{x}, \boldsymbol{y})$ of Definition 7.4 and its gradient, is computable by a polynomial-time Turing Machine. We prove Theorem 7.6 through a series of Lemmas. To simplify notation we set $b \triangleq \log 1/\varepsilon$.

Definition D.1. For a $x \in \mathbb{R}$, we denote by $[x]_b \in \mathbb{R}$, a value represented by the b bits such that

$$|[x]_b - x| \leq 2^{-b}.$$

Lemma D.2. *There exist Turing Machines M_{S_∞}, $M_{S'_\infty}$ that given input $x \in [0,1]$ and ε in binary form, compute $[S_\infty(x)]_b$ and $[S'_\infty(x)]_b$ in time polynomial in $b = \log(1/\varepsilon)$ and the binary representation of x.*

Proof. The Turing Machine M_{S_∞} outputs the fist b bits of the following quantity,

$$W(x) = \left[\frac{1}{1 + \left[2^{\left[-\frac{1}{x} + \frac{1}{x-1} \right]_{b'}} \right]_{b'}} \right]_{b'}$$

where b' will be selected sufficiently large. Notice it is possible to compute the above quantity due to the fact that all functions $\frac{1}{\gamma} + \frac{1}{\gamma-1}$, 2^{γ} and $\frac{1}{1+\gamma}$ can be computed with accuracy $2^{-b'}$ in polynomial time with respect to b' and the binary representation of γ [Bre76]. Moreover,

$$\begin{aligned}
&\left|\left[\frac{1}{1+\left[2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}\right]_{b'}}\right]_{b'} - \frac{1}{1+2^{-\frac{1}{x}+\frac{1}{x-1}}}\right| \\
&\quad\le \left|\left[\frac{1}{1+\left[2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}\right]_{b'}}\right]_{b'} - \frac{1}{1+\left[2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}\right]_{b'}}\right| \\
&\qquad\quad + \left|\frac{1}{1+\left[2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}\right]_{b'}} - \frac{1}{1+2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}}\right| \\
&\qquad\quad + \left|\frac{1}{1+2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}} - \frac{1}{1+2^{-\frac{1}{x}+\frac{1}{x-1}}}\right| \\
&\quad\le 2^{-b'} + \left|\left[2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}\right]_{b'} - 2^{\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'}}\right| \\
&\qquad\quad + \ln 2\left|\left[-\frac{1}{x}+\frac{1}{x-1}\right]_{b'} - \left(-\frac{1}{x}+\frac{1}{x-1}\right)\right| \\
&\quad\le 4\cdot 2^{-b'}
\end{aligned}$$

where the first inequality follows from triangle inequality and the second follows from the facts that $1/(1+\gamma)$ is a 1-Lipschitz function of γ for $\gamma \ge 0$, and $1/(1+2^{\gamma})$ is an $\ln(2)$-Lipschitz function of γ for $\gamma \ge 0$. The last inequality follows from the definition of $[\cdot]_{b'}$. Hence $W(x)$ is indeed equal to $[S_{\infty}(x)]_b$ if we choose $b' = b+2$.

Next we explain how $M_{S'_{\infty}}$ computes $[S'_{\infty}(x)]_b$. First notice that $S'_{\infty}(x)$ is equal to

$$S'_{\infty}(x) = \ln 2 \cdot \frac{\frac{1}{x^2}2^{-\frac{1}{x}+\frac{1}{x-1}} - \frac{1}{(x-1)^2}2^{-\frac{1}{x}+\frac{1}{x-1}}}{\left(2^{-\frac{1}{x}} + 2^{\frac{1}{x-1}}\right)^2}.$$

To describe how to compute $S'_{\infty}(x)$ we first assume that we have computed the following quantities. Then based on these quantities we show how $S'_{\infty}(x)$ can be computed and finally we consider the computation of these quantities.

- $\triangleright$ $[\ln 2]_{b'}$,
- $\triangleright$ $A \leftarrow \left[\frac{1}{x^2}2^{-\frac{1}{x}+\frac{1}{x-1}}\right]_{b'}$,
- $\triangleright$ $B \leftarrow \left[\frac{1}{(x-1)^2}2^{-\frac{1}{x}+\frac{1}{x-1}}\right]_{b'}$,
- $\triangleright$ $C \leftarrow \left[\left(2^{-\frac{1}{x}} + 2^{\frac{1}{x-1}}\right)^2\right]_{b'}$.

Then $M_{S'_\infty}$ outputs the fist b bits of the quantity $\left[[\ln 2]_{b'} \cdot \left[\frac{A+B}{C}\right]_{b'}\right]_{b'}$. We now prove that

$$\left| [\ln 2]_{b'} \left[\frac{A+B}{C}\right]_{b'} - \underbrace{\ln 2 \frac{A+B}{C}}_{S'_\infty(x)} \right| \leq \Theta\left(2^{-b'}\right)$$

Consider the function $g(\alpha, \beta, \gamma) = \frac{\alpha+\beta}{\gamma}$ where $|\alpha|, |\beta| \leq c_1$ and $|\gamma| \geq c_2$ where c_1, c_2 are universal constants. Notice that $g(\alpha, \beta, \gamma)$ is c-Lipschitz for $c = \sqrt{\frac{2}{c_2^2} + \frac{2c_1}{c_2^2}}$. Since for sufficiently large b' all the quantities $|A|, |B|, \left|\frac{1}{x^2} 2^{-\frac{1}{x}+\frac{1}{x-1}}\right|, \left|\frac{1}{(x-1)^2} 2^{-\frac{1}{x}+\frac{1}{x-1}}\right| \leq c_1$ and $|C|, \left(2^{-\frac{1}{x}} + 2^{\frac{1}{x-1}}\right)^2 \geq c_2$ where c_1, c_2 are universal constants we get that

$$\left| \left[\frac{A+B}{C}\right]_{b'} - \frac{A+B}{C} \right| \leq \Theta\left(2^{-b'}\right).$$

Now consider the function $g(\alpha, \beta) = \alpha \cdot \beta$ where $|\alpha|, |\beta| \leq c$ where c is a universal constant. In this case $g(\alpha, \beta)$ is $\sqrt{2}c$-Lipschitz continuous. Since for b' sufficiently large all the quantities $|[\ln 2]_{b'}|, \left|\left[\frac{A+B}{C}\right]_{b'}\right|, \ln 2, \left|\frac{A+B}{C}\right|$ are bounded by a universal constant c, we have that,

$$\left| [\ln 2]_{b'} \left[\frac{A+B}{C}\right]_{b'} - \ln 2 \frac{A+B}{C} \right| \leq \Theta\left(2^{-b'}\right)$$

Next we explain how the values A, B and C are computed while $[\ln(2)]'_b$ can easily be computed via standard techniques [Bre76].

- **Computation of A.** The Turing Machine $M_{S'_\infty}$ will compute A by taking the first b' bits of the following quantity,
$$\left[2^{\left[-\frac{1}{x}+\frac{1}{x-1}+2\ln x/\ln 2\right]_{b''}}\right]_{b''}$$
where b'' will be taken sufficiently large. We remark that both where both the exponentiation and the natural logarithm can be computed in polynomial-time with respect to the number of accuracy bits and the binary representation of the input [Bre76]. The function $\frac{1}{x^2} 2^{-\frac{1}{x}+\frac{1}{x-1}} = 2^{-\frac{1}{x}+\frac{1}{x-1}+2\ln x/\ln 2}$ is c-Lipschitz where c is a universal constant. Thus,
$$\left| \left[2^{\left[-\frac{1}{x}+\frac{1}{x-1}+2\ln x/\ln 2\right]_{b''}}\right]_{b''} - \frac{1}{x^2} 2^{-\frac{1}{x}+\frac{1}{x-1}} \right| \leq \Theta(2^{-b''}).$$
- **Computation of B.** Using the same arguments as for A.
- **Computation of C.** To compute C we first compute b'' bits of the following quantity,
$$\left[\frac{1}{\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''} + \left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}}\right]_{b''}^2$$
We first argue that
$$\left| \left[\frac{1}{\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''} + \left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}}\right]_{b''}^2 - \left(\frac{1}{2^{-\frac{1}{x}} + 2^{\frac{1}{x-1}}}\right)^2 \right| \leq \Theta\left(2^{-b''}\right)$$
The latter follows by applying the triangle inequality and the following 3 inequalities.

1.

$$\left|\left[\frac{1}{\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''}+\left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}}\right]_{b''}^{2}-\left(\frac{1}{\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''}+\left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}}\right)^{2}\right| \leq \Theta(2^{-b''})$$

this holds since for $b'' > 1$ we have

$$\left[\frac{1}{\left(\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''}+\left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}\right)}\right]_{b''} \quad \text{and} \quad \frac{1}{\left(\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''}+\left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}\right)}$$

are both upper-bounded by 2 while the function $g(\alpha) = \alpha^2$ is 4-Lipschitz for $|\alpha| \leq 2$.

2.

$$\left|\left(\frac{1}{\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''}+\left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}}\right)^{2}-\left(\frac{1}{2^{-\left[\frac{1}{x}\right]_{b''}}+2^{\left[\frac{1}{x-1}\right]_{b''}}}\right)^{2}\right| \leq \Theta\left(2^{-b''}\right)$$

The latter follows since for b'' larger than a universal constant, both $\left[2^{-\left[\frac{1}{x}\right]_{b''}}\right]_{b''} + \left[2^{\left[\frac{1}{x-1}\right]_{b''}}\right]_{b''}$ and $2^{-\left[\frac{1}{x}\right]_{b''}} + 2^{\left[\frac{1}{x-1}\right]_{b''}}$ are greater than a universal constant c, while the function $g(\alpha, \beta) = 1/(\alpha+\beta)^2$ is $\Theta\left(c^3\right)$-Lipschitz for $\alpha + \beta \geq c$.

3.

$$\left|\left(\frac{1}{2^{-\left[\frac{1}{x}\right]_{b''}}+2^{\left[\frac{1}{x-1}\right]_{b''}}}\right)^{2}-\left(\frac{1}{2^{-\frac{1}{x}}+2^{\frac{1}{x-1}}}\right)^{2}\right| \leq \Theta\left(2^{-b''}\right)$$

The latter follows since for b'' larger than a universal constant it holds that both the quantities in the left hand side are greater than a positive universal constant c, while the function $g(\alpha, \beta) = 1/(2^{-\alpha} + 2^{\beta})$ for $2^{-\alpha} + 2^{\beta} \geq c$, $\alpha \geq 0$, and $\beta \leq 0$ is $\Theta\left(1/c^3\right)$-Lipschitz.

This concludes the proof of the lemma. □

Lemma D.3. *There exist Turing Machines M_Q and $M_{Q'}$ that given $\boldsymbol{x} \in [0,1]^d$ and $\varepsilon > 0$ in binary form, respectively compute $[Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x})]_b$ and $[\nabla Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x})]_b$ for all vertices $\boldsymbol{v} \in ([N]-1)^d$ with $Q_{\boldsymbol{v}}^{\boldsymbol{c}}(\boldsymbol{x}) > 0$, where $b = \log(1/\varepsilon)$. These vertices are most $d+1$. Moreover both M_Q and $M_{Q'}$ run in polynomial time with respect to b, d and the binary representation of $\boldsymbol{x}$.*

Proof. Both M_Q, $M_{Q'}$ firsts compute the canonical representation $p_{\boldsymbol{x}}^{\boldsymbol{c}} \in [0,1]^d$ with the respect to the cell $R(\boldsymbol{x})$ in which $\boldsymbol{x}$ lies. Such a cell $R(\boldsymbol{x})$ can be computed by taking the first $(\log N + 1)$-bits at each coordinate of $\boldsymbol{x}$. The source vertex $\boldsymbol{s}^{\boldsymbol{c}} = (s_1, \ldots, s_d)$ and the target vertex $\boldsymbol{t}^{\boldsymbol{c}} = (t_1, \ldots, t_d)$ with respect to $R(\boldsymbol{x})$ are also computed. Once this is done we are only interested in vertices $\boldsymbol{v} \in R_{\boldsymbol{c}}(\boldsymbol{x})$ for which

$$p_\ell > p_j \quad \text{for all } \ell \in A_{\boldsymbol{v}}^{\boldsymbol{c}}, j \in B_{\boldsymbol{v}}^{\boldsymbol{c}}$$

since for all the other $\boldsymbol{v} \in ([N]-1)^d$ both $Q_{\boldsymbol{v}}^c(\boldsymbol{x}) = 0$ and $\nabla Q_{\boldsymbol{v}}^c(\boldsymbol{x}) = 0$. These vertices, that are denoted by $R_+(\boldsymbol{x})$, are at most $d+1$ and can be computed in polynomial time.

The vertices $\boldsymbol{v} \in R_+(\boldsymbol{x})$ can be computed in polynomial time as follows: **(i)** the coordinates $p_1, \ldots, p_d$ are sorted in increasing order **ii)** for each $m = 0, \ldots, d$ compute the vertex $\boldsymbol{v}^m \in R_{\boldsymbol{c}}(\boldsymbol{x})$,

$$\boldsymbol{v}_j^m = \begin{cases} s_j & \text{if coordinate } j \text{ belongs in the first } m \text{ coordinates wrt the order of } \boldsymbol{p}_{\boldsymbol{x}}^c \\ t_j & \text{if coordinate } j \text{ belongs in the last } d-m \text{ coordinates wrt the order of } \boldsymbol{p}_{\boldsymbol{x}}^c \end{cases}$$

By Definition 8.7 it immediately follows that $R_+(\boldsymbol{x}) \subseteq \bigcup_{m=0}^d \{\boldsymbol{v}^m\}$ which also establish that $|R_+(\boldsymbol{x})| \leq d+1$.

Once $R_+(\boldsymbol{x})$ is computed, M_Q computes for each pair $(\ell, j) \in B_{\boldsymbol{v}}^c \times A_{\boldsymbol{v}}^c$ the value of the number $\left[S_\infty(S(p_\ell) - S(p_j))\right]_{b'}$ for some accuracy b' that we determine later but depends polynomially on b, d and the input accuracy of $\boldsymbol{x}$. Then each $\boldsymbol{v} \in R_+(\boldsymbol{x})$, M_Q outputs as $[Q_{\boldsymbol{v}}^c(\boldsymbol{x})]_b$ the fist b bits of the following quantity

$$\left[\prod_{\ell \in B_{\boldsymbol{v}}^c, j \in A_{\boldsymbol{v}}^c} \left[S_\infty(S(p_\ell) - S(p_j))\right]_{b'}\right]_{b'}$$

where b' is selected sufficiently large. We next prove that this computation indeed outputs $[Q_{\boldsymbol{v}}^c(\boldsymbol{x})]_b$ accurately.

To simplify notation let $S_\infty(S(p_\ell) - S(p_j))$ be denoted by $S_{\ell j}$, $A_{\boldsymbol{v}}^c$ denoted by A and $B_{\boldsymbol{v}}^c$ denoted by B. Then,

$$\begin{aligned} \left|\left[\Pi_{\ell \in B, j \in A} \left[S_{\ell j}\right]_{b'}\right]_{b'} - \Pi_{\ell \in B, j \in A} S_{\ell j}\right| &\leq \left|\left[\Pi_{\ell \in B, j \in A} \left[S_{\ell j}\right]_{b'}\right]_{b'} - \Pi_{\ell \in B, j \in A} \left[S_{\ell j}\right]_{b'}\right| \\ &+ \left|\Pi_{\ell \in B, j \in A} \left[S_{\ell j}\right]_{b'} - \Pi_{\ell \in B, j \in A} S_{\ell j}\right| \\ &\leq 2^{-b'} + \left|\Pi_{\ell \in B, j \in A} \left[S_{\ell j}\right]_{b'} - \Pi_{\ell \in B, j \in A} S_{\ell j}\right| \end{aligned}$$

Consider the function $g(\boldsymbol{y}) = \prod_{\ell \in B, j \in A} y_{\ell j}$. For $\boldsymbol{y} \in [0, 1 + 1/d^2]^{|A| \times |B|}$, $\|\nabla g(\boldsymbol{y})\|_2 \leq \Theta(d)$. As a result, for all $\boldsymbol{y}, \boldsymbol{z} \in [0, 1 + 1/d^2]^{|A| \times |B|}$,

$$|g(\boldsymbol{y}) - g(\boldsymbol{z})| \leq \Theta(d) \cdot \left[\sum_{\ell \in B, j \in A} (y_{\ell j} - z_{\ell j})\right]^{1/2}$$

In case the accuracy $b' \geq \Theta(\log d)$ then $\left[S_{\ell j}\right]_{b'} \leq S_{\ell j} + 1/d^2 \leq 1 + 1/d^2$ and the above inequality applies. Thus,

$$\begin{aligned} \left|\prod_{\ell \in B, j \in A} \left[S_{\ell j}\right]_{B'} - \Pi_{\ell \in B, j \in A} S_{\ell j}\right| &\leq \Theta(d) \left[\sum_{\ell \in B, j \in A} \left(\left[S_{\ell j}\right]_{B'} - S_{\ell j}\right)\right]^{1/2} \\ &\leq \Theta(d^2) \cdot 2^{-b'} \end{aligned}$$

Overall, $\left|\left[\Pi_{\ell \in B, j \in A} \left[S_{\ell j}\right]_{b'}\right]_{b'} - \Pi_{\ell \in B, j \in A} S_{\ell j}\right| \leq \Theta(d^2) \cdot 2^{-b'}$ which concludes the proofof the corrected of $[Q_{\boldsymbol{v}}^c(\boldsymbol{x})]_b$ by selecting $b' = b + \Theta(\log d)$.

In order to compute $\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_\ell}$ where $\ell \in B_v^c$ (symmetrically for $j \in A_v^c$), $M_{Q'}$ additionally computes the $\left[S'_\infty(S(p_\ell) - S(p_j))\right]_{b'}$ with accuracy b'. To simplify notation we denote with $S'_\infty(S(p_\ell) - S(p_j))$ with $S'_{\ell j}$ and $S'(p_i)$ by S'_i. Then $M_{Q'}$ outputs,

$$\left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i}\right]_{b'} \leftarrow \left[\frac{1}{t_i - s_i} \cdot \left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right]_{b'}\right]_{b'}$$

$$\text{where} \quad \left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right]_{b'} \leftarrow \left[\sum_{j \in A} \left[S'_{ij}\right]_{b'} \cdot \left[S'_i\right]_{b'} \Pi_{m \in A/j, \ell \in B} \left[S_{\ell m}\right]_{b'}\right]_{b'}$$

Observe that $t_i - s_i = \frac{\text{sign}(t_i - s_i)}{N-1}$ and thus $\frac{1}{t_i - s_i} \cdot \left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right]_{b'}$ can be exactly computed. We next prove that these computations of $\left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i}\right]_{b'}$ and $\left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right]_{b'}$ are correct.

We first bound $\left|\left[S'_{ij}\right]_{b'} \cdot \left[S'_i\right]_{b'} \cdot \Pi_{m \in A/\{j\}, \ell \in B} \left[S_{\ell m}\right]_{b'} - S'_{ij} \cdot S'_i \cdot \Pi_{m \in A/\{j\}, \ell \in B} S_{\ell m}\right|$.

Consider the function $g(y_1, y_2, \boldsymbol{y}) = y_1 \cdot y_2 \cdot \prod_{m \in A/\{j\}, \ell \in B} y_{\ell m}$. As previously done, for $y_1, y_2 \in [0, 6]$ and $\boldsymbol{y} \in [0, 1 + 1/d^2]^{|A| \times |B| - 1}$ we have that, $\|\nabla g(y_1, y_2, \boldsymbol{y})\|_2 \leq \Theta(d)$. If $b' \leq \Theta(\log d)$ then $\left|S'_{ij}\right|, S'_i \leq 6$ and $S_{\ell m} \in [0, 1 + 1/d^2]$. As a result,

$$\left|\left[S'_{ij}\right]_{b'} \cdot \left[S'_i\right]_{b'} \cdot \Pi_{m \in A/\{j\}, \ell \in B} \left[S_{\ell m}\right]_{b'} - S'_{ij} \cdot S'_i \cdot \Pi_{m \in A/\{j\}, \ell \in B} S_{\ell m}\right| \quad \leq \quad \Theta(d^2) \cdot 2^{-b'}.$$

We can now use the above inequality to bound $\left|\left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right]_{b'} - \frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right|$. More precisely,

$$\begin{aligned}
&\left|\left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right]_{b'} - \frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right| \\
&\quad \leq 2^{-b} + \left|\sum_{j \in A} \left[S'_{ij}\right]_{b'} \cdot \left[S'_i\right]_{b'} \cdot \prod_{m \in A/\{j\}, \ell \in B} \left[S_{\ell m}\right]_{b'} - \sum_{j \in A} S'_{ij} \cdot S'_i \cdot \prod_{m \in A/\{j\}, \ell \in B} S_{\ell m}\right| \\
&\quad \leq \Theta(d^3) \cdot 2^{-b'}
\end{aligned}$$

We finally get that

$$\left|\left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i}\right]_{b'} - \frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i}\right| \leq 2^{-b'} + N \left|\left[\frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right]_{b'} - \frac{\partial Q_v^c(\boldsymbol{x})}{\partial p_i}\right| \leq \Theta(Nd^3) \cdot 2^{-b'}.$$

Thus the analysis is completed by selecting $b' = b + \Theta(\log d) + \Theta(\log N)$. □

Lemma D.4. *There exist Turing Machines M_P and $M_{P'}$ that given $\boldsymbol{x} \in [0, 1]^d$ and $\varepsilon > 0$ in binary form compute $[\mathrm{P}_v(\boldsymbol{x})]_b$ and $[\nabla \mathrm{P}_v(\boldsymbol{x})]_b$ respectively for all vertices $v \in ([N] - 1)^d$ with $\mathrm{P}_v(\boldsymbol{x}) > 0$, where $b = \log(1/\varepsilon)$. These vertices are most $d + 1$. Moreover both M_P and $M_{P'}$ run in polynomial time with respect to b, d and the binary representation of $\boldsymbol{x}$.*

Proof. M_P first runs M_Q of Lemma D.3 to find the coefficients $Q_v^c(\boldsymbol{x}) > 0$. We remind that these vertices are denoted with $R_+(\boldsymbol{x})$ and $|R_+(\boldsymbol{x})| \leq d + 1$. Then for each $v \in R_+(\boldsymbol{x})$, M_P outputs as $[\mathrm{P}_v(\boldsymbol{x})]_b$ the fist b bits of the quantity,

$$\left[\frac{[Q_v^c(x)]_{b'}}{\sum_{v' \in R_+(\boldsymbol{x})} \left[Q_{v'}^c(x)\right]_{b'}}\right]_{b'}$$

where we determine the value of b' later in the proof but it is chosen to be polynomial in b and d. We next present the proof that the above expression correctly computes $[\mathrm{P}_v(\boldsymbol{x})]_b$.
For accuracy $b' \geq \Theta(d^2 \log d)$ we get that,

$$\begin{aligned}
\sum_{v' \in R_+(\boldsymbol{x})} [Q^c_{v'}(x)]_{b'} &\geq \sum_{v' \in R_+(\boldsymbol{x})} Q^c_{v'}(x) - \Theta(d) \cdot 2^{-b'} \\
&= \sum_{v' \in R_c(\boldsymbol{x})} Q^c_{v'}(x) - \Theta(d) \cdot 2^{-b'} \\
&\geq \Theta\left(1/d\right)^{d^2}\Big) - \Theta(d) \cdot 2^{-b'} \\
&\geq \Theta\left((1/d)^{d^2}\right)
\end{aligned}$$

Consider the function $g(\boldsymbol{y}) = y_i/(\sum_{j=1}^{d+1} y_j)$. Notice that for $\boldsymbol{y} \in [0,1]^{d+1}$ and $\sum_{j=1}^{d+1} y_j \geq \mu$ then $\|\nabla g(\boldsymbol{y})\|_2 \leq \Theta(d^{3/2}/\mu^2)$. The latter implies that for $\boldsymbol{y}, \boldsymbol{z} \in [0,1]^{d+1}$ such that $\sum_{j=1}^{d+1} y_j \geq \mu$ and that $\sum_{j=1}^{d+1} z_j \geq \mu$, it holds that

$$\left| \frac{y_i}{\sum_{j=1}^{d+1} y_j} - \frac{z_i}{\sum_{j=1}^{d+1} z_j} \right| \leq \Theta\left(\frac{d^{3/2}}{\mu^2}\right) \cdot \|\boldsymbol{y} - \boldsymbol{z}\|_2 .$$

Since there are at most $d+1$ vertices $\boldsymbol{v}' \in R_+(\boldsymbol{x})$ while both the term $\sum_{v' \in R_+(\boldsymbol{x})} \left[Q^c_{v'}(\boldsymbol{x})\right]_{b'}$ and the term $\sum_{v' \in R_+(\boldsymbol{x})} Q^c_{v'}(\boldsymbol{x})$ are greater than $\Theta\left((1/d)^{d^2}\right)$, we can apply the above inequality with $\mu = \Theta\left((1/d)^{d^2}\right)$ and we get the following

$$\begin{aligned}
&\left| \frac{[Q^c_v(x)]_{b'}}{\sum_{v' \in R_+(\boldsymbol{x})} \left[Q^c_{v'}(x)\right]_{b'}} - \frac{Q^c_v(x)}{\sum_{v' \in R_+(\boldsymbol{x})} Q^c_{v'}(x)} \right| \\
&\qquad \leq \Theta\left(d^{2d^2+3/2}\right) \cdot \left[\sum_{v' \in R_+(\boldsymbol{x})} \left([Q^c_{v'}(x)]_{b'} - Q^c_{v'}(x) \right)^2 \right]^{1/2} \\
&\qquad \leq \Theta\left(d^{2d^2+2}\right) \cdot 2^{-b'}
\end{aligned}$$

Overall, we have that

$$\begin{aligned}
&\left| \left[\frac{[Q^c_v(x)]_{b'}}{\sum_{v' \in R_+(\boldsymbol{x})} \left[Q^c_{v'}(x)\right]_{b'}} \right]_{b'} - \frac{Q^c_v(x)}{\sum_{v' \in R_c(\boldsymbol{x})} Q^c_{v'}(x)} \right| \\
&\qquad \leq \left| \left[\frac{[Q^c_v(x)]_{b'}}{\sum_{v' \in R_+(\boldsymbol{x})} \left[Q^c_{v'}(x)\right]_{b'}} \right]_{b'} - \frac{[Q^c_v(x)]_{b'}}{\sum_{v' \in R_+(\boldsymbol{x})} \left[Q^c_{v'}(x)\right]_{b'}} \right| \\
&\qquad\quad + \left| \frac{[Q^c_v(x)]_{b'}}{\sum_{v' \in R_+(\boldsymbol{x})} \left[Q^c_{v'}(x)\right]_{b'}} - \frac{Q^c_v(x)}{\sum_{v' \in R_+(\boldsymbol{x})} Q^c_{v'}(x)} \right| \\
&\qquad \leq \Theta\left(d^{2d^2+1}\right) 2^{-b'}
\end{aligned}$$

The proof is completed via selecting $b' = b + \Theta(d^2 \log d)$.

In order to compute $\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial x_i}$ the Turing machine $M_{P'}$ computes all vertices $R_+(\boldsymbol{x})$ the coefficients $\frac{\partial Q_v^c(\boldsymbol{x})}{\partial x_i}$ with accuracy b'. Then for each $\boldsymbol{v} \in R_+(\boldsymbol{x})$ the Turing Machine $M_{P'}$ outputs,

$$\left[\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial x_i}\right]_{b'} \leftarrow \left[\frac{1}{t_i - s_i} \cdot \left[\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial p_i}\right]_{b'}\right]_{b'}$$

$$\text{where} \quad \left[\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial p_i}\right]_{b'} \leftarrow \left[\frac{\left[\frac{\partial Q_v(\boldsymbol{x})}{\partial p_i}\right]_{b'} \cdot \sum_{\boldsymbol{v}' \in R_+(\boldsymbol{x})} [Q_{\boldsymbol{v}'}(\boldsymbol{x})]_{b'} - [Q_v(\boldsymbol{x})]_{b'} \cdot \sum_{\boldsymbol{v}' \in R_+(\boldsymbol{x})} \left[\frac{\partial Q_{\boldsymbol{v}'}(\boldsymbol{x})}{\partial p_i}\right]_{b'}}{\left(\sum_{\boldsymbol{v}' \in R_+(\boldsymbol{x})} [Q_{\boldsymbol{v}'}(\boldsymbol{x})]_{b'}\right)^2}\right]_{b'}$$

Similarly as above and as in Lemma D.3 we can prove that if $b' \geq b + \Theta(d^2 \log d) + \Theta(\log N)$, $\left|\left[\frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial p_i}\right]_{b'} - \frac{\partial \mathrm{P}_v(\boldsymbol{x})}{\partial p_i}\right| \leq 2^{-b}$. □

Proof of Theorem 7.6. Let $R(\boldsymbol{x})$ be the cell at which $\boldsymbol{x}$ lies. The Turing Machine $M_{f_{\mathcal{C}_l}}$ initially calculates the vertices $\boldsymbol{v} \in R_c(\boldsymbol{x})$ with coefficient $\mathrm{P}_{\boldsymbol{v}}(\boldsymbol{x}) > 0$. We remind that this set is denoted by $R_+(\boldsymbol{x})$ and $|R_+(\boldsymbol{x})| \leq d+1$. Then $M_{f_{\mathcal{C}_l}}$ outputs the first b bits of the following quantity,

$$\left[f_{\mathcal{C}_l(\boldsymbol{x},\boldsymbol{y})}\right]_{b'} = \sum_{j=1}^{d} [\alpha(\boldsymbol{x},j)]_{b'} \cdot (x_j - y_j) \quad \text{where} \quad [\alpha(\boldsymbol{x},j)]_{b'} = \sum_{\boldsymbol{v}' \in R_+(\boldsymbol{x})} \mathcal{C}_l(\boldsymbol{v},j) \cdot [\mathrm{P}_{\boldsymbol{v}}(\boldsymbol{x})]_{b'}$$

we next prove that the above computation is correct.

$$\begin{aligned}
\left|\left[f_{\mathcal{C}_l(\boldsymbol{x},\boldsymbol{y})}\right]_{b'} - f_{\mathcal{C}_l(\boldsymbol{x},\boldsymbol{y})}\right| &= \left|\sum_{j=1}^{d} [\alpha(\boldsymbol{x},j)]_{b'} \cdot (x_j - y_j) - \sum_{j=1}^{d} \alpha(\boldsymbol{x},j) \cdot (x_j - y_j)\right| \\
&\leq \sum_{j=1}^{d} |[\alpha(\boldsymbol{x},j)] - \alpha(\boldsymbol{x},j)| \\
&= \sum_{j=1}^{d} \left|\sum_{\boldsymbol{v}' \in R_+(\boldsymbol{x})} \mathcal{C}_l(\boldsymbol{v},j) \cdot [\mathrm{P}_{\boldsymbol{v}}(\boldsymbol{x})]_{b'} - \sum_{\boldsymbol{v}' \in R_+(\boldsymbol{x})} \mathcal{C}_l(\boldsymbol{v},j) \cdot \mathrm{P}_{\boldsymbol{v}}(\boldsymbol{x})\right| \\
&\leq \sum_{j=1}^{d} \sum_{\boldsymbol{v}' \in R_+(\boldsymbol{x})} |[\mathrm{P}_{\boldsymbol{v}}(\boldsymbol{x})]_{b'} - \mathrm{P}_{\boldsymbol{v}}(\boldsymbol{x})| \\
&\leq d \cdot (d+1) \cdot 2^{-b'}
\end{aligned}$$

Setting $b' = b + \Theta(\log d)$ we get the desired result. Similarly for $\frac{\partial f_{\mathcal{C}_l(\boldsymbol{x},\boldsymbol{y})}}{\partial x_i}$ and $\frac{\partial f_{\mathcal{C}_l(\boldsymbol{x},\boldsymbol{y})}}{\partial y_i}$. □

E Convergence of PGD to Approximate Local Minimum

In this section we present for completeness the folklore result that the Projected Gradient Descent with convex projection set converges fast to a first order stationary point. Using the same ideas that we presented in Section 5 this result implies that Projected Gradient Descent solves the LocalMin problem in time $\mathrm{poly}(1/\varepsilon, L, G, d)$ when (ε, δ) in the input are in the local regime. Also observe that although the following proof assumes access to the exact value of the gradient ∇f it is very simple to adapt the proof to the case where we only have access to ∇f with accuracy ε^3. We leave this as an exercise to the reader.

Theorem E.1. *Let $f : K \to \mathbb{R}$ be an L-smooth function and $K \subseteq \mathbb{R}^d$ be a convex set. The projected gradient descent algorithm started at $\boldsymbol{x}_0$, with step size η, after at most $T \geq \frac{2L(f(\boldsymbol{x}_0)-f(\boldsymbol{x}^\star))}{\varepsilon^2}$ steps outputs a point $\hat{\boldsymbol{x}}$ such that*

$$\|\hat{\boldsymbol{x}} - \Pi_K\left(\hat{\boldsymbol{x}} - \eta \nabla f(\hat{\boldsymbol{x}})\right)\|_2 \leq \eta \cdot \varepsilon$$

where $\eta = 1/L$ and $\boldsymbol{x}^\star$ is a global minimum of f.

Proof. If we run the Projected Gradient Descent algorithm on f then we have

$$\boldsymbol{x}_{t+1} \leftarrow \Pi_K\left(\boldsymbol{x}_t - \eta \nabla f(\boldsymbol{x}_t)\right)$$

then due to the L-smoothness of f we have that

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) + \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle + \frac{L}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2^2.$$

We can now apply Theorem 1.5.5 (b) of [FP07] to get that

$$\langle \eta \cdot \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle \leq -\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2^2 \implies$$

$$\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle \leq -\frac{1}{\eta} \cdot \|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2^2$$

If we combine these then we have that

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \left(\frac{1}{\eta} - \frac{L}{2}\right)\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2^2.$$

So if we pick $\eta = 1/L$ then we get

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \frac{L}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2^2.$$

If sum all the above inequalities and divide by T then we get

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2^2 \leq \frac{2}{T \cdot L}\left(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_T)\right)$$

which implies that

$$\min_{0 \leq t \leq T-1}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2 \leq \sqrt{\frac{2}{T \cdot L}\left(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_T)\right)}$$

Therefore for $T \geq \frac{2L(f(\boldsymbol{x}_0)-f(\boldsymbol{x}^\star))}{\varepsilon^2}$ we have that

$$\min_{0 \leq t \leq T-1}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|_2 \leq \eta \cdot \varepsilon = \varepsilon/L.$$

□