Mathematical Programming (2025) 213:281–341
https://doi.org/10.1007/s10107-024-02139-3

FULL LENGTH PAPER

Series A

The computational complexity of finding stationary points in non-convex optimization

Alexandros Hollender[1] · **Manolis Zampetakis**[2]

Received: 24 January 2024 / Accepted: 12 August 2024 / Published online: 27 September 2024

Abstract

Finding approximate stationary points, i.e., points where the gradient is approximately zero, of non-convex but smooth objective functions f over unrestricted d-dimensional domains is one of the most fundamental problems in classical non-convex optimization. Nevertheless, the computational and query complexity of this problem are still not well understood when the dimension d of the problem is independent of the approximation error. In this paper, we show the following computational and query complexity results:

1. The problem of finding approximate stationary points over unrestricted domains is PLS-complete.
2. For $d = 2$, we provide a zero-order algorithm for finding ε-approximate stationary points that requires at most $O(1/\varepsilon)$ value queries to the objective function.
3. We show that any algorithm needs at least $\Omega(1/\varepsilon)$ queries to the objective function and/or its gradient to find ε-approximate stationary points when $d = 2$. Combined with the above, this characterizes the query complexity of this problem to be $\Theta(1/\varepsilon)$.
4. For $d = 2$, we provide a zero-order algorithm for finding ε-KKT points in constrained optimization problems that requires at most $O(1/\sqrt{\varepsilon})$ value queries to the objective function. This closes the gap between works of Bubeck and Mikulincer and Vavasis and characterizes the query complexity of this problem to be $\Theta(1/\sqrt{\varepsilon})$.
5. Combining our results with a recent result of Fearnley et al., we show that finding approximate KKT points in constrained optimization is reducible to finding approximate stationary points in unconstrained optimization but the converse is impossible.

An extended abstract appeared in the proceedings of COLT 2023.

✉ Alexandros Hollender
alexandros.hollender@cs.ox.ac.uk

Manolis Zampetakis
emmanouil.zampetakis@yale.edu

[1] University of Oxford, Oxford, UK

[2] Yale University, New Haven, USA

1 Introduction

One of the most fundamental problems in optimization is the following unconstrained problem which arises in the whole spectrum of scientific research and engineering and is an essential part of modern machine learning systems [3, 16]

$$\min_{x \in \mathbb{R}^d} f(x). \tag{1}$$

Depending on the structure of f, the above optimization problem takes different forms. In this paper, we focus on the case where f is possibly non-convex, it has Lipschitz gradients, i.e., is *smooth*, and its value is lower bounded. This formulation of (1) corresponds to one of the standard formulations of optimization problems in non-convex optimization. The global optimization of (1) is intractable even if we allow approximation errors [15]. Following the classical paradigm, we consider the problem of finding *approximate stationary points (or ε-stationary points)* of (1), i.e., we want to find $x^\star \in \mathbb{R}^d$ such that

$$\left\| \nabla f\left(x^\star\right) \right\|_2 \leq \varepsilon \qquad \text{where} \quad \varepsilon > 0. \tag{2}$$

For this problem, it is well-known that gradient descent with appropriate but constant step size converges to an ε-stationary point $x^\star$ after $O(1/\varepsilon^2)$ iterations, if we assume that f has constant smoothness, e.g., see [16].

Despite the significance of this problem, essentially no lower bounds were known until recently when [7] showed that if $d > 1/\varepsilon^2$ then any algorithm that only has query access to f and ∇f needs time $\Omega(1/\varepsilon^2)$ to find an ε-stationary point. This breakthrough result settles the query complexity of finding approximate stationary points when the number of dimensions grows faster that $1/\varepsilon^2$.

In this paper our goal is to understand the complexity of finding stationary points when the condition $d \geq 1/\varepsilon^2$ does not hold. In particular, we want to answer the following two questions;

Question 1: *What is the computational complexity of finding ε-stationary points for any $d > 1$?*

Question 2: *What is the query complexity of finding approximate stationary points when $d = 2$?*

We note that for $d = 1$, the problem can be solved using only $O(\log(1/\varepsilon))$ function value queries [8]. There has also been a lot of interest in the constrained version of the problem (1), which corresponds to the following constrained optimization task

$$\min_{x \in [0,1]^d} f(x). \tag{3}$$

For this problem, the corresponding notion of approximate stationary points is finding *approximate KKT points*. From the definition of stationary points and KKT points, it is not clear if finding stationary points in unconstrained optimization or finding KKT

points in constrained optimization is harder, since the solution concepts are different. For this constrained problem we know the following:

- [11]: finding approximate KKT points is CLS-complete.
- [18]: finding ε-approximate KKT points in $d = 2$ requires at least $\Omega(1/\sqrt{\varepsilon})$ value and gradient queries.
- [4]: there exist an algorithm for finding ε-approximate KKT points in $d = 2$ that requires at most $O((1/\sqrt{\varepsilon})\sqrt{\log(1/\varepsilon)})$ value and gradient queries.

Although the above results draw an almost complete picture for answering Questions 1 and 2 in the constrained optimization setting the following two questions still remain open.

Question 3: *Is there an algorithm with complexity $O(1/\sqrt{\varepsilon})$ for finding ε-approximate KKT points in constrained optimization for $d = 2$?*

Question 4: *What is the relationship between finding approximate KKT points in constrained optimization and finding approximate stationary points in unconstrained optimization?*

In this paper we provide a complete answer for Questions 1 and 2 in unconstrained optimization and Questions 3 and 4 for constrained optimization.

1.1 Our results

Our two main results are the following:

1. PLS**-completeness (Theorem 3.1).** We show that finding ε-stationary points of smooth and bounded functions in unconstrained optimization is PLS-complete for any $d > 1$.
2. **Unconstrained algorithm for** $d = 2$ **(Theorem 4.1).** For $d = 2$ we provide a zero-order algorithm that finds ε-stationary points of smooth and bounded functions in unconstrained optimization using at most $O(1/\varepsilon)$ value queries to the objective. This result resolves a recent open problem of [8].

Once we establish these results we show that our PLS-completeness result has the following important corollaries.

(a) **Query lower bound for** $d = 2$**. (Theorem 3.2).** We show that any algorithm for finding ε-stationary points of smooth and bounded functions in unconstrained optimization when $d = 2$ needs at least $\Omega(1/\varepsilon)$ value and/or gradient queries.

(b) **Constrained vs Unconstrained optimization (Corollary 1).** It is possible to reduce finding approximate KKT points in constrained optimization to finding approximate stationary points in unconstrained optimization but the converse is impossible.

Additionally, the techniques that we use to show our second main result have the following corollary.

(c) **Constrained algorithm for $d = 2$ (Theorem 4.2).** For $d = 2$ we provide a zero-order algorithm that finds ε-KKT points of smooth and bounded functions in constrained optimization using at most $O(1/\sqrt{\varepsilon})$ value queries to the objective. This result closes the gap between the upper bound of [4] and the lower bound of [18].

The above results resolve our Questions 1 - 4. In particular: result 1. resolves Question 1, the combination of 2. and (a) resolve Question 2, (b) resolves Question 4, and (c) closes the gap of Question 3.

A particularly surprising result, at least for the authors, is that (b) provides the relationship between finding stationary points in unconstrained optimization and finding KKT points in constrained optimization. It is almost a reflex to believe that constrained optimization is harder than unconstrained optimization but the notion of stationary points in unconstrained optimization is provably harder than the notion of KKT points in constrained optimization. We can see this in two ways. First, finding approximate stationary points is PLS-complete as per Theorem 3.1, whereas finding approximate KKT points is CLS-complete as per [11] and CLS $\subseteq$ PLS. Second, finding approximate KKT points is reducible to finding approximate stationary points in a black-box way, but the converse is in fact impossible.

We believe that our results significantly increase our understanding of the complexity of finding stationary points in unconstrained optimization. Furthermore, we depart from the classical methods of showing lower bounds in optimization that often use the resisting oracle technique and we give one more example where tools from complexity theory can be utilized to provide new lower bounds in classical optimization problems, e.g, [1, 10, 11].

1.2 Open questions

The following couple of open questions arise from our work:

1. Characterize the query complexity of the problem (both in its unconstrained and constrained versions) in fixed dimension $d \geq 3$. In particular, any progress in dimension $d = 3$ is interesting.
2. Characterize the query and computational complexity of higher order methods, both in the unconstrained and in the constrained setting.

2 Preliminaries

Notation. We define $[[n, m]]$ for $n \leq m$, to be the set of all integers between n and m. We use e_i to represent the ith unit vector in the standard basis of $\mathbb{R}^d$. We use $\ell(x, y)$ to denote the line segment between two points $x, y \in \mathbb{R}^d$. For any number $z \in \mathbb{R}$ we use $\text{len}(z)$ to denote the number of bits in the binary representation of z. We can extend the domain of len to vectors as follows: let $x \in \mathbb{R}^d$, then $\text{len}(x) = \sum_{i=1}^{d} \text{len}(x_i)$.

Our main object of study is a real-valued, non-convex, but smooth function $f : \mathbb{R}^d \to \mathbb{R}$. We assume that we have access to f in two different ways:

- **Black Box Model.** This is the classical model used in optimization theory where we assume that for every point $x \in \mathbb{R}^d$ in the space there is a way to evaluate $f(x)$ and/or $\nabla f(x)$. If we only have access to $f(x)$ then we say that we have *zero-order black box access* to f, whereas if we have access to both $f(x)$ and $\nabla f(x)$ we say that we have *first-order black box access* to f.
- **White Box Model.** This is the classical model used in complexity theory to characterize the computational complexity of optimization problems. In this model we are given the description of a polynomial-time Turing machine $\mathcal{C}_f$ that computes $f(x)$ and $\nabla f(x)$. More precisely, given some input $x \in \mathbb{R}^d$, described using b bits, $\mathcal{C}_f$ runs in time upper bounded by some polynomial in b and outputs approximate values for $f(x)$ and $\nabla f(x)$. 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 Section 2 of [10] for more details about how to formally study the computational complexity of problems that take as input a polynomial-time Turing machine.

Lipschitzness, Smoothness, and Normalization. Our main objects of study are continuously differentiable Lipschitz, smooth, and bounded functions $f : \mathbb{R}^d \to \mathbb{R}$. A continuously differentiable function f is called L-*Lipschitz* if $|f(x) - f(y)| \leq L\|x - y\|_2$, for all $x, y \in \mathbb{R}^d$, L-*smooth* if $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$, for all $x, y \in \mathbb{R}^d$, and B-*bounded* is $|f(x)| \leq B$ for all $x \in \mathbb{R}^d$. In the black box model we will assume that the function satisfies these properties. In the white box model we will allow for violation solutions to handle cases where the properties are not satisfied.

Approximate Stationary points. As in classical unconstrained non-convex optimization, given the function $f : \mathbb{R}^d \to \mathbb{R}$ our goal is to find a point $x^\star$ such that $\|\nabla f(x^\star)\|_2 \leq \varepsilon$ for some given parameter $\varepsilon > 0$. We call such a point $x^\star$ an ε-*stationary point of* f.

In this paper we have two tasks: (1) to characterize the computational complexity of finding an ε-stationary point when f is given in the white box model, and (2) find tight upper and lower bounds on the number of oracle calls to f that are needed when $d = 2$ and f is given in the black box model. For each one of these results we need some definitions.

2.1 Complexity of total search problems and the classes PLS and CLS

An important property of finding stationary points of a function f is that if f is bounded and smooth then there always exists an ε-stationary point of f with bit representation poly $\log(B, L, 1/\varepsilon)$, as we will see in Sect. 3.1. From this property, it is clear that the correct complexity landscape to study this problem is that of total search problems that is captured by the subclasses of the class TFNP which we define in Appendix A. In particular, we are interested in the complexity classes PLS as defined in [9, 13]. PLS, which stands for Polynomial Local Search, is defined as the set of all TFNP problems that reduce in polynomial time to the following problem.

Definition 2.1 LOCALOPT:
Input: Circuits $N, P : [2^n] \to [2^n]$ (N is the neighbor function and P the potential function).
Goal: Find $v \in [2^n]$ such that $P(N(v)) \geq P(v)$.

In this paper, we make use of the following total search problem that is known to also characterize the complexity class PLS [14]. See also Appendix A.1 for more details.

Definition 2.2 ITER:
Input: Boolean circuit $C : [2^n] \to [2^n]$ with $C(1) > 1$.
Goal: Find v such that either

- $C(v) < v$, or
- $C(v) > v$ and $C(C(v)) = C(v)$.

Next we define the computational problem that we are interested in for the white box model.

Definition 2.3 STATIONARY:
Input:

- precision parameter $\varepsilon > 0$,
- Turing machines $\mathcal{C}_f$ and $\mathcal{C}_{\nabla f}$ representing $f : \mathbb{R}^d \to \mathbb{R}$ and $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$,
- a boundedness constant $B > 0$, and a smoothness constant $L > 0$.

Goal: Find $x^\star \in \mathbb{R}^d$ such that $\|\nabla f(x^\star)\|_2 \leq \varepsilon$.

Alternatively, we also accept one of the following violations as a solution:

- $x, y \in \mathbb{R}^d$ that violate the Lipschitzness or smoothness of f,
- $x \in \mathbb{R}^d$ such that $|f(x)| > B$,
- $x, y \in \mathbb{R}^d$ that certify that $\mathcal{C}_{\nabla f}$ computes ∇f incorrectly.

Violations and promise-preserving reductions. There is no known way of syntactically enforcing the Turing machines $\mathcal{C}_f, \mathcal{C}_{\nabla f}$ to be Lipschitz-continuous and bounded. Thus, to ensure that our problem indeed lies in TFNP, we also allow solutions witnessing violations of these properties. This "trick" was also used by [9] for the definition of CLS. Nevertheless, we note that our hardness result for STATIONARY also applies to the promise version of the problem, where we are promised that the input satisfies the properties.

2.2 Query complexity in the black box model

In the black box model we are only interested in the query complexity of the problem. This means that we are interested in understanding how many queries to f and ∇f an algorithm has to make before it finds an ε-stationary point. Fix a deterministic algorithm $\mathcal{A}$, an initialization $x_0 \in \mathbb{R}^d$, and a bounded, Lipschitz and smooth function f. We define $\mathcal{T}(\mathcal{A}, x_0; f)$ to be the number of queries that the deterministic algorithm $\mathcal{A}$ has to make to f and ∇f before it outputs an ε-stationary point of f.

Remark 1 (*Query lower bound for randomized algorithms.*) In this paper we focus for simplicity on proving query lower bounds for deterministic algorithms. Nevertheless, we want to mention that the techniques of Section 6 of [4] can be used to show that our Theorem 3.2 holds even for randomized algorithms with a polylogarithmic loss in the query complexity.

3 PLS-completeness of finding stationary points in unconstrained optimization

In this section we characterize the computational complexity of the optimization problem StationaryPoint. The machinery that we use to prove Theorem 3.1 can be also utilized to show the tight query lower bound for StationaryPoint in 2-dimensions, see Theorem 3.2.

Theorem 3.1 *The problem* StationaryPoint *is* PLS*-complete, for any* $d \geq 2$. *The* PLS*-hardness of* StationaryPoint *holds even if we are promised that the input satisfies the boundedness, the Lipschitzness, and the smoothness properties. Finally, our hardness reduction is black-box preserving.*

If we combine Theorem 3.1 with the CLS-completeness result of finding KKT points in constrained optimization we get the following very interesting corollary for the relation between stationary points in constrained and unconstrained optimization.

Corollary 1 *There is an efficient black-box reduction from finding approximate KKT points in constrained optimization to finding approximate stationary points in unconstrained optimization. On the other hand, there is no efficient black-box reduction from finding approximate stationary points in unconstrained optimization to finding approximate KKT points in constrained optimization.*

Proof This is an immediate corollary of the following three results: Theorem 3.1, the CLS completeness of finding approximate KKT points of [11], and the black-box separation PLS $\nsubseteq$ CLS, which follows from PLS $\nsubseteq$ PPAD [5, 6, 12]. □

In Sect. 3.1 we show that StationaryPoint is in PLS, and in Sect. 3.2 that it is PLS-hard. Finally, in Sect. 3.3 we show a black-box query complexity result that we get from Theorem 3.1 when $d = 2$.

3.1 Membership in PLS

The idea for showing the membership of StationaryPoint in PLS is very similar to the well-known argument that gradient descent finds and ε-stationary point in $O(B \cdot L/\varepsilon^2)$ steps. The argument is that if we start from a point x such that $\|\nabla f(x)\|_2 \geq \varepsilon$ and we apply a gradient descent step with step size $1/L$ then the function value $f(x')$ in the new point x' has to be at least ε^2/L smaller than $f(x)$. Since f is B-bounded this can only happen $O(B \cdot L/\varepsilon^2)$ times and hence within this number of steps gradient descent has to find an ε-stationary point.

The above argument suggests that the value $f(x)$ can be used as a potential function and the gradient descent update as a neighboring function to show the membership of StationaryPoint in PLS. The issue with that is that our domain is unbounded and hence it is not possible at first glance to define a finite domain which would be necessary in order to provide a reduction to LocalOpt. Of course, taking a closer look at the gradient descent argument, we observe that we can provide an upper bound on the distance R that gradient descent needs to travel before it reaches an ε-stationary point. Unfortunately, this is still not enough since truncating all the points that have distance more than R will create fallacious solutions on the boundary.

To resolve this we need one more idea: focus only on points whose distance from the initial point and their value suggest that they could be points visited by gradient descent starting from the origin. In this way we are able to show that there are no ad-hoc solutions created on the boundary, which proves the following lemma.

Lemma 1 *It holds that* StationaryPoint$\in$ PLS.

Proof We formalize the argument that we presented in the beginning of the section. Let $\gamma = \varepsilon/(\sqrt{d}L)$, $m = \lceil 8\sqrt{d}B/(\varepsilon\gamma)\rceil$, and $R = m\gamma$. Define the following grid of $[-R, R]^d$

$$G_\gamma = \left\{\gamma \cdot a \mid a \in [[-m, m]]^d\right\},$$

In other words, G_γ is the ortho-canonical grid of $[-R, R]^d$ with step γ between two neighboring vertices of the grid.

Our next goal is to define a neighbor function $N : G_\gamma \to G_\gamma$ and a potential function $P : G_\gamma \to [-B-1, B+1]$ such that every point $x \in G_\gamma$ with $P(N(x)) \geq P(x)$ is an ε-stationary point of f.

Valid and invalid points. From the definition of StationaryPoint we are given a Turing machine $\mathcal{C}_f$ that on input x runs in time poly(len(x)) and outputs the value of $f(x)$. First, we define the set of *valid points* $\mathcal{V}_\gamma$ which is a subset of the grid points G_γ. A point $v \in G_\gamma$ is valid, i.e., $v \in \mathcal{V}_\gamma$, if and only if, $f(v) \leq f(0) - (\varepsilon/(2\sqrt{d})) \cdot \|v\|_2$. Intuitively, a point v is valid if and only if it is possible for the gradient descent flow starting at 0 to reach v without passing through any $\varepsilon/(2\sqrt{d})$-stationary point.[1] We define the set of *invalid points* $\mathcal{I}_\gamma$ to be $\mathcal{I}_\gamma = G_\gamma \backslash \mathcal{V}_\gamma$. Finally, for every $v \in G_\gamma$ we

[1] Note that this does not mean that this is the case, just that it is possible. Conversely, if a point v is invalid, then it is impossible for the gradient flow starting at 0 to visit v without encountering an $\varepsilon/(2\sqrt{d})$-stationary point.

define $\Gamma(v)$ to be the set of *immediate neighbors* of v as follows

$$\Gamma(v) = \left\{u \in G_\gamma \mid u \neq v,\ \|u - v\|_2 \leq \gamma\right\}.$$

It is easy to see that $|\Gamma(v)| \leq 2d$, where we have exact equality for all the points of the grid in the interior of the box $[-R, R]^d$ and inequality for the points on the boundary. From the definition of valid points we have the following claim.

Claim 1 *For all $v \in \mathcal{V}_\gamma$ it holds that $|\Gamma(v)| = 2 \cdot d$. (Or v yields a counter-example to the boundedness of f.)*

Proof To show this we only need to argue that all the points of G_γ that lie on the boundary of $[-R, R]^d$ are invalid. To see this observe that any point w on the boundary of $[-R, R]^d$ satisfies that $\|w\|_2 \geq R$. Therefore, for w to be valid we need to have that $f(w) \leq f(0) - (\varepsilon/(2\sqrt{d}))R$ but $f(0) \leq B$. This means that we need $f(w) \leq B - (\varepsilon/(2\sqrt{d}))R$ which, since $R \geq 8\sqrt{d}B/\varepsilon$ by definition, implies $f(w) \leq B - 4B = -3B$. But if $f(w) \notin [-B, B]$, then w can be used as a violation solution, since it violates the boundedness of f. □

The potential function P. We start by defining the function P. For the valid points we define $P(v)$ to be the output of the Turing machine $\mathcal{C}_f$ on v, i.e., $P(v) = f(v)$. For any invalid point $v \in \mathcal{I}_\gamma$, we define $P(v) = B + 1$.

The neighbor function N. For every $v \in \mathcal{I}_\gamma$ we define $N(v) = 0$. Since $P(v) = B+1$ and 0 is valid by definition, this means that $P(0) < P(v)$ and hence none of the invalid points can satisfy $P(N(v)) \geq P(v)$. Next we define the function N on the valid points. We define $N(v)$ to be the immediate neighbor of v that has minimum value of $P(v)$, i.e.,

$$N(v) = \operatorname*{argmin}_{w \in \Gamma(v)} P(w).$$

It remains to show that any solution of the LocalOpt instance with inputs P, N as defined above, yields a solution to the Stationary problem. Let $v \in G_\gamma$ be such that $\|\nabla f(v)\|_2 > \varepsilon$. We will show that v cannot be a solution of the LocalOpt instance. In other words, we will show that $P(N(v)) < P(v)$. We have already noted above that if v is invalid, it cannot be a solution to LocalOpt. Thus, we may also assume that v is valid. Since $\|\nabla f(v)\|_2 > \varepsilon$, there exist $i \in [d]$ and $s \in \{0, 1\}$ such that $\langle \nabla f(v), (-1)^s e_i \rangle \leq -\varepsilon/\sqrt{d}$. Consider the point $w = v + \gamma \cdot (-1)^s e_i$. Note that by Claim 1, $w \in \Gamma(v)$, i.e., w is an immediate neighbor of v. If Taylor's theorem does not hold for v, w, then we have found a violation solution. If, on the other hand, it

holds, then we can write

$$\begin{aligned} f(w) \le f(v) + \langle \nabla f(v), w - v\rangle + \frac{L}{2}\|w-v\|_2^2 &= f(v) + \gamma\langle \nabla f(v), (-1)^s e_i\rangle \\ &\quad + \frac{L\gamma^2}{2}\|(-1)^s e_i\|_2^2 \\ &\le f(v) - \frac{\varepsilon\gamma}{\sqrt{d}} + \frac{L\gamma^2}{2} \\ &= f(v) - \frac{\varepsilon\gamma}{2\sqrt{d}} \\ &= f(v) - \frac{\varepsilon}{2\sqrt{d}}\|w-v\|_2 \end{aligned}$$

where we used $\gamma = \varepsilon/(\sqrt{d}L)$. From this inequality we can already deduce that $f(w) < f(v)$. Furthermore, using the fact that v is valid, and thus $f(v) \le f(0) - \varepsilon/(2\sqrt{d}) \cdot \|v\|_2$, we also deduce that $f(w) \le f(0) - \varepsilon/(2\sqrt{d}) \cdot \|w\|_2$. In other words, w is also valid and thus $P(w) = f(w) < f(v) = P(v)$. It remains to show that $P(N(v)) \le P(w)$. But this follows from the definition of N and the fact that $w \in \Gamma(v)$.

To conclude the reduction we need to transform N, P from Turing machines to boolean circuits and their domain and range to $[2^n]$ for some $n \in \mathbb{N}$. First, observe that the binary representation of the points in $v \in G_\gamma$ is $\text{len}(v) \le d \cdot \log(R/\gamma)$. Hence, the output $f(v)$ for any $v \in G_\gamma$ has $\text{len}(f(v)) = q(\text{len}(v))$ where q is some polynomial that is specified together with the description of $\mathcal{C}_f$. We can then pick $n = q(\text{len}(v))$ and we can map the set $[2^n]$ to describe both G_γ and the possible outputs of the Turing machine P that we described above. This way we can make N, P to be mappings from $[2^n]$ to itself and have running time $r(n)$ for some polynomial r. We can now use classical transformations of Turing machines with running time $r(n)$ to boolean circuits with size $r(n)\log(r(n))$. This completes the reduction of StationaryLocalOpt. Based on the above, it is also easy to see that the reduction is polynomial-time in the input, i.e., the size of the Turing machines $\mathcal{C}_f$, $\mathcal{C}_{\nabla f}$, and polylogarithmic in the parameters B, L, and $1/\varepsilon$. □

Remark 2 Note that we in fact proved that StationaryLocalOpt is in PLS even when we are only given the Turing machine $\mathcal{C}_f$, and we do not require a Turing machine $\mathcal{C}_{\nabla f}$ that computes the gradient, assuming that we have a promise that the smoothness of f holds. Moreover, for the PLS-membership we do not need the boundedness assumption $|f(x)| \le B$ but we only need that $f(0) - f(x) \le B$ for all $x \in \mathbb{R}^d$, where 0 can be replaced by any fixed initial point x_0.

3.2 PLS-hardness

In this section we show that the problem StationaryLocalOpt is PLS-hard even when the number of dimensions is $d = 2$. For $d = 1$ finding stationary points can be done in

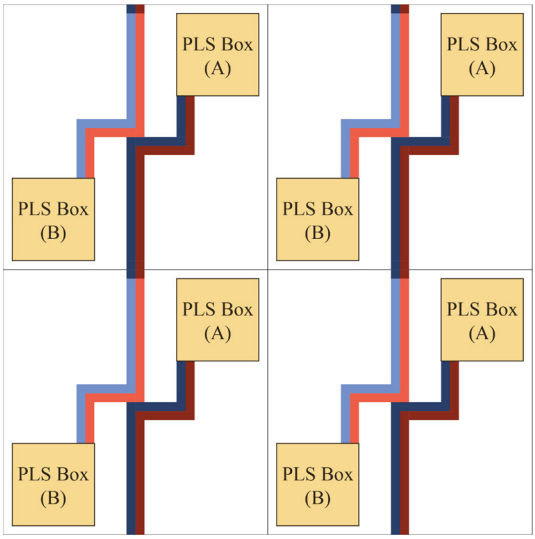


Fig. 1 High-level structure of the construction of f in our hardness result of Lemma 2

$O(\log(LB/\varepsilon))$ as was shown in [8]. Hence, our result implies that for $d \geq 2$ there is no $O(\text{poly}\log(LB/\varepsilon))$ algorithm for solving STATIONARY unless FP = PLS.

Lemma 2 *The problem* STATIONARY *is* PLS*-hard. Moreover, the* PLS*-hardness of* STATIONARY *holds even if we are promised that the boundedness, the Lipschitzness, and the smoothness of the input of* STATIONARY *hold. Finally, our hardness reduction is black-box preserving.*

We begin with a high-level description of our construction and mention the key components of our proof.

High-level Proof Sketch of Lemma 2. We construct f periodically, i.e., we define a function g over a square $[0, M]^2$ and then to get f we repeat copies of g in the whole plane $\mathbb{R}^2$ as shown in Fig. 1. If g satisfies some boundary properties then this repetition of g yields a bounded and smooth function f. So, we can now focus on the construction of g.

The function g is a continuously differentiable function that is defined over $[0, M]^2$ which means that it will attain a minimum at some point $x_{\min}$ and a maximum at some point $x_{\max}$. Obviously we have to have that $\nabla g(x_{\min}) = 0$ and $\nabla g(x_{\max}) = 0$. Hence we want to ensure two things: (1) that $x_{\min}$ and $x_{\max}$ are in places that correspond to solutions of an ITER instance, and (2) all stationary points correspond to local minima and local maxima. For this, we place two boxes in the construction of g: the PLS Box (A) and the PLS Box (B) (see Fig. 1). The PLS Box (A) is the only place where local minima can be formed and the PLS Box (B) is the only place where local maxima can be formed. We also want to make sure that no solution can arise in the space outside PLS Box (A) and PLS Box (B). To do that we use a structure shown in Fig. 1 where the blue colors correspond to low function values, the red colors to high function values and the background has colors between blue and red. Both blue and red values are

decreasing as each one of the coordinates increase we move up and to the right and the background decreases as we go from left to right. Hence:

- ▷ If we start from any point in the background and we are looking for a local minimum then we will move to the right until we hit the blue region. Once we are inside the blue region we will move up and to the right until we are inside the PLS Box (A).
- ▷ Similarly, if we start from any point in the background and we are looking for a local maximum then we will move to the left until we hit the red region. Once we are inside the red region we will move down and to the left until we are inside the PLS Box (B).

The above observation gives an intuition about why local minima can appear only in PLS Box (A) and local maxima only in PLS Box (B) in our construction as well as about why we should not expect any stationary points outside of these PLS boxes. We formally show these properties about the space outside the PLS boxes in Sect. 3.2.4.

PLS Boxes. This is where we use the ITER instance.[2] At a very high level, starting from an ITER instance we move the blue-red in the bottom left corner of PLS Box (A) following the paths of the circuit C and in this way we make sure that when the blue-red line ends, which is the only place where a local minimum can be formed, will be in places where the ITER problem with input C has a solution. The construction of PLS Box (B) is similar except that the orientation of the box is rotated by 180^o. We give the formal construction of the PLS boxes in Sect. 3.2.5.

In the above sketch we skipped an important part of our proof which is how to construct this piecewise function while satisfying smoothness of f. For this we use bi-cubic interpolation techniques (see Sect. 3.2.3) and we need to make sure that all the regions that we construct satisfy the conditions to apply bi-cubic interpolation without creating stationary points in places other than local minima and local maxima.

Full proof of Lemma 2. In the remainder of this section we provide the full proof of Lemma 2. We prove the hardness result by constructing a function $f : \mathbb{R}^2 \to \mathbb{R}$ where finding approximate stationary points is PLS-complete. For $d > 2$ we can use the function f below and define a function $h : \mathbb{R}^d \to \mathbb{R}$ as $h(x) = f(x_1, x_2)$ and it is easy to see that finding approximate stationary points for h is as hard as finding approximate stationary points for f. Hence, our hardness holds for any $d \geq 2$.

We describe the hardness construction in a top-down way. We start with how we use some *tiles* to cover the whole plane $\mathbb{R}^2$, then we describe the structure of these tiles, and how they contain what we call a **PLS Box** and then we describe the structure of the PLS Box. To complete our proof we need to make sure that we do not create any stationary points in places that do not correspond to a solution of the PLS instance that we want to solve. This last step is not too complicated, but rather tedious because it involves checking many small parts of our construction.

[2] Our PLS boxes have many similarities with the PLS Labyrinths of [11]. Nevertheless, our construction is more challenging because of a different background that we need to use since we define f on the whole plane.

3.2.1 Periodic structure of *f*

We need to define the function f over the whole plane $\mathbb{R}^2$ and we want to make sure that the value of f remains bounded. In order to satisfy this property, we define f as a periodic function. More formally we will design some function $g : [0, M]^2 \to \mathbb{R}$ for some **odd**[3] natural number M and then we define f as:

$$f(x, y) = g\Big(x - M \cdot \Big\lfloor \frac{x}{M} \Big\rfloor, y - M \cdot \Big\lfloor \frac{y}{M} \Big\rfloor\Big) \tag{4}$$

where $\lfloor x \rfloor$ denotes the integer part of x and hence $x - M\lfloor \frac{x}{M} \rfloor$ represents the "modulo M" operator over the reals. Of course, in order for this construction to be correct we need to make sure that on the boundary of $[0, M]^2$ the different copies of the function g touch in a consistent way. In particular, we need that

$$g(x, 0) = g(x, M) \quad \text{and} \quad g(0, y) = g(M, y) \quad \text{for all } x, y \in [0, M]. \tag{5}$$

$$\nabla g(x, 0) = \nabla g(x, M) \quad \text{and} \quad \nabla g(0, y) = \nabla g(M, y) \quad \text{for all } x, y \in [0, M]. \tag{6}$$

3.2.2 Defining *g* on a grid

The next step is to design the function g. Because we start from a combinatorial problem, i.e., ITER, and our goal is to construct a continuous function it is very helpful to design g in two steps: first we define the values and the gradients of g on a discrete grid that is a subset of the square $[0, M]^2$ and then we use an interpolation, described in Sect. 3.2.3, to define g in the rest of $[0, M]^2$. The grid that we use is the following:

$$G_M = \{(a, b) \mid a, b \in \mathbb{N}, a, b \in [0, M]\} \triangleq [[0, M]]^2. \tag{7}$$

Our construction starts with defining function values $g(a, b)$ and gradient values $\nabla g(a, b)$ for the points $(a, b) \in G_M$ of the grid G_M. Outside the grid, the values of g are defined via interpolation as we describe in the next section.

3.2.3 Bi-cubic interpolation

Given the function and gradient values of g on G_M we apply *bi-cubic interpolation* in every *small box* of G_M to define g everywhere in $[0, M]^2$. Consider a small box $[a, a + 1] \times [b, b + 1]$, with $(a, b) \in G_M$. If we have the function and the gradient values of g for all the four corners (a, b), $(a, b + 1)$, $(a + 1, b)$, $(a + 1, b + 1)$ then we

[3] We need to pick M odd to make sure that there is an even number of natural numbers in the interval $[0, M]$.

can define g in every point of the small box $[a, a+1] \times [b, b+1]$ using a polynomial of the form:

$$g(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} \cdot (x-a)^i \cdot (y-b)^j \tag{8}$$

where the coefficients a_{ij} are computed as follows

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{9}$$
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} g(a,b) & g(a,b') & g_y(a,b) & g_y(a,b') \\ g(a',b) & g(a',b') & g_y(a',b) & g_y(a',b') \\ g_x(a,b) & g_x(a,b') & 0 & 0 \\ g_x(a',b) & g_x(a',b') & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

where $a' := a+1$, $b' := b+1$, and where g_x and g_y denote the partial derivatives of g with respect to x and y respectively. It is well-known that the above interpolation yields a function that is bounded, Lipschitz, and smooth, e.g., see [17]. The bound on the function value and the Lipschitz constant are determined from the specified function and gradient values on the corner of the corresponding cell as we will see in Sect. 3.2.6.

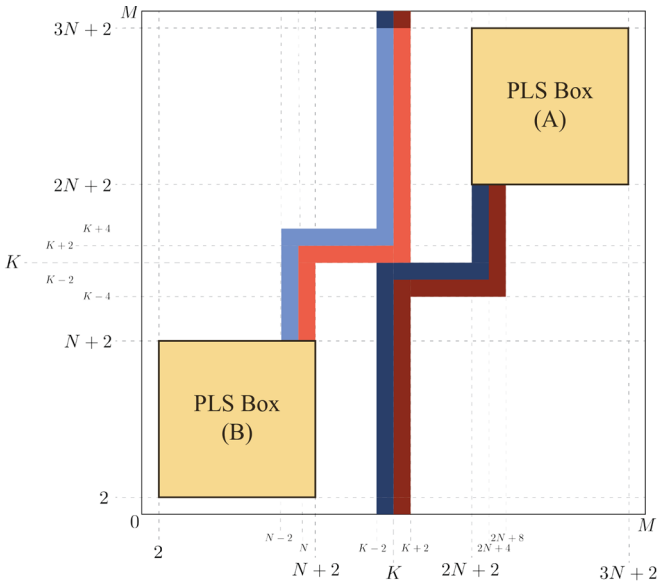
From now on we will focus on the definition of g on the grid points G_M and when we finish the description of this we will come back to the properties of bi-cubic interpolation to complete our construction.

3.2.4 Structure of the function g on G_M

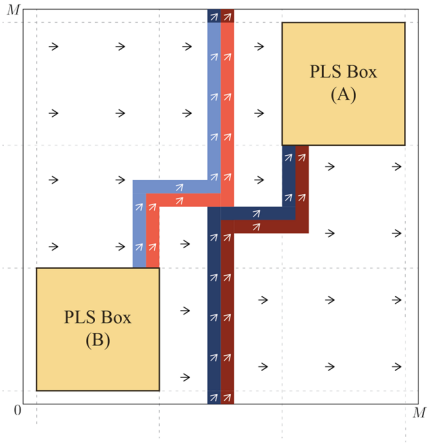
As we explained we pick M to be an odd number so that in the interval $[0, M]$ there is an even number of natural numbers. We define $K = M/2$ (observe that K is not an integer!) and $M = 3N + 4$ or in other words $N = (M-4)/3$. For now we will keep N as a parameter and we will pick a specific value of N when we construct the PLS instance inside g.

The definition of g on G_M contains **eight** different regions. We represent 5 of them with different colors, one of them corresponds to the top 2-rows of G_M and is there to satisfy the boundary conditions 5, and we defer the definition of the last two for Sect. 3.2.5. For each colored region we specify the following thing: (a) its location, (b) how the function values are defined in the discrete points of G_M in this region, and (c) how the gradient values are defined in the discrete points of G_M in this region. The values of the gradient that we use are either $(-\delta, 0)$ or $(0, -\delta)$ with $\delta = 1/2$. We use the vector multiplier δ here because it makes some arguments easier.

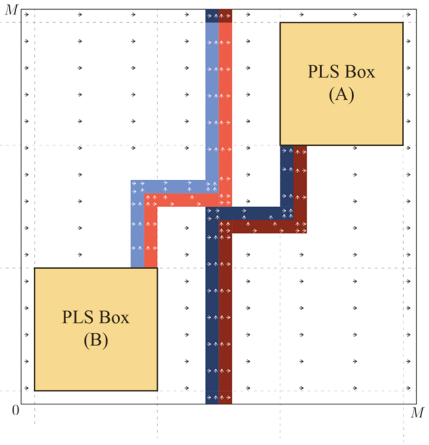
To be more precise, our construction involves the following regions: **dark blue region**, **dark red region**, **light blue region**, **light red region**, **background**, **top region**, **PLS Box (A)**, and **PLS Box (B)** as shown in Fig. 2a. Our goal is to first define all the regions except the PLS Boxes and show that these regions do not contain any stationary



(a) The locations of the eight regions that we use to define g on G_M.



(b) The arrows indicate the direction in which the function values decrease in every region except the PLS boxes.



(c) The arrows indicate the direction of the negative gradient in every region except the PLS boxes.

Fig. 2 The location, direction that the function values decrease and directions of the negative gradient for the regions that define g on G_M

point. Then, in Sect. 3.2.5, we define the structure of the PLS boxes and how we can encode a PLS instance in them. e finally then show that solutions can occur only in places that correspond to solutions of the initial PLS instance.

Dark Blue Region. We denote this region by $\mathcal{D}_B$ and has the following specifications:

(a) Location. As shown in Fig. 2a, the dark blue region contains the following three segments of points, each of which as we can see has width 2, two of these are

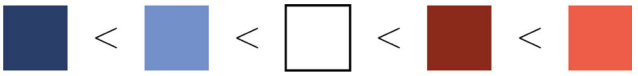


Fig. 3 The relative order of function values in different regions. The middle white color with black boundary corresponds to the function values of the background region

vertical segments and one is horizontal. In particular, $\mathcal{D}_B$ contains the following points:

- ▷ The points $(a, b) \in G_M$ such that $K - 2 \leq a \leq K$ and $0 \leq b \leq K$.
- ▷ The points $(a, b) \in G_M$ such that $K - 2 \leq a \leq 2N + 4$ and $K - 2 \leq b \leq K$.
- ▷ The points $(a, b) \in G_M$ such that $2N + 2 \leq a \leq 2N + 4$ and $K \leq b \leq 2N + 2$.

(b) Function Values. We define the following function

$$h_{DB}(x, y) = -x - y - 6 \cdot M \tag{10}$$

and for every $(a, b) \in \mathcal{D}_B$ we set $g(a, b) = h_{DB}(a, b)$. As shown in Fig. 2b the function values inside the dark blue region decrease as both of the coordinates x and y increase. Another property of dark blue region, that will be clear once we define all the regions, is that it has the smallest values among all the regions of g as shown in Fig. 3.

(c) Gradient Values. The gradient values in the dark blue region is either $(-\delta, 0)$ or $(0, -\delta)$. As we explained in (a) the dark blue region contains two vertical segments and one horizontal segment each of which has length 2. For the vertical segments the left part has gradient $(-\delta, 0)$ and the right $(0, -\delta)$. For the horizontal segment the top part has gradient $(-\delta, 0)$ and the bottom $(0, -\delta)$. More precisely we have that:

- ▶ For the points $(a, b) \in \mathcal{D}_B$ it holds that $\nabla g(a, b) = (-\delta, 0)$ in the following cases:
  - ▷ $a = K - 3/2$ and $0 \leq b \leq K$,
  - ▷ $K - 2 \leq a \leq 2N + 3$ and $b = K - 1/2$,
  - ▷ $a = 2N + 5/2$ and $K \leq b \leq 2N + 2$.
- ▶ For the points $(a, b) \in \mathcal{D}_B$ it holds that $\nabla g(a, b) = (0, -\delta)$ in the following cases:
  - ▷ $a = K - 1/2$ and $0 \leq b \leq K - 1$,
  - ▷ $K - 1 \leq a \leq 2N + 4$ and $b = K - 3/2$,
  - ▷ $a = 2N + 7/2$ and $K - 1 \leq b \leq 2N + 2$.

In Fig. 2c we show the direction of the negative gradient for all the regions.

Dark Red Region. We denote this region by $\mathcal{D}_R$ and has the following specifications:

(a) Location. As shown in Fig. 2a, the dark red region contains the following three segments of points, each of which as we can see as width 2, two of these are vertical segments and one is horizontal. In particular, $\mathcal{D}_R$ contains the following points:

- ▷ The points $(a, b) \in G_M$ such that $K \leq a \leq K + 2$ and $0 \leq b \leq K - 2$.

- ▹ The points $(a, b) \in G_M$ such that $K \leq a \leq 2N + 4$ and $K \leq b \leq K + 2$.
- ▹ The points $(a, b) \in G_M$ such that $2N + 4 \leq a \leq 2N + 6$ and $K - 4 \leq b \leq 2N + 2$.

(b) Function Values. We define the following function

$$h_{DR}(x, y) = -x - y + 3 \cdot M \tag{11}$$

and for every $(a, b) \in \mathcal{D}_R$ we set $g(a, b) = h_{DR}(a, b)$. As shown in Fig. 2b the function values inside the dark red region decrease as both of the coordinates x and y increase. Another property of dark red region, that will be clear once we define all the regions, is that it has the value greater than both of the blue regions and the background and has value smaller that the light red region as shown in Fig. 3.

(c) Gradient Values. The gradient values in the dark red region is either $(-\delta, 0)$ or $(0, -\delta)$. As we explained in (a) the dark red region contains two vertical segments and one horizontal segment each of which has length 2. For the vertical segments the left part has gradient $(0, -\delta)$ and the right $(-\delta, 0)$. For the horizontal segment the top part has gradient $(0, -\delta)$ and the bottom $(-\delta, 0)$. More precisely we have that:

- ▶ For the points $(a, b) \in \mathcal{D}_R$ it holds that $\nabla g(a, b) = (-\delta, 0)$ in the following cases:
  - ▹ $a = K + 3/2$ and $0 \leq b \leq K - 3$,
  - ▹ $K + 1 \leq a \leq 2N + 5$ and $b = K - 7/2$,
  - ▹ $a = 2N + 11/2$ and $K - 4 \leq b \leq 2N + 2$.
- ▶ For the points $(a, b) \in \mathcal{D}_R$ it holds that $\nabla g(a, b) = (0, -\delta)$ in the following cases:
  - ▹ $a = K + 1/2$ and $0 \leq b \leq K - 2$,
  - ▹ $K \leq a \leq 2N + 5$ and $b = K - 5/2$,
  - ▹ $a = 2N + 9/2$ and $K - 3 \leq b \leq 2N + 2$.

In Fig. 2c we show the direction of the negative gradient for all the regions.

Light Blue Region. We denote this region by $\mathcal{L}_B$ and has the following specifications:

(a) Location. As shown in Fig. 2a, the light blue region contains the following three segments of points, each of which as we can see as width 2, two of these are vertical segments and one is horizontal. In particular, $\mathcal{L}_B$ contains the following points:

- ▹ The points $(a, b) \in G_M$ such that $K - 2 \leq a \leq K$ and $K + 2 \leq b \leq M$.
- ▹ The points $(a, b) \in G_M$ such that $N - 2 \leq a \leq K$ and $K + 2 \leq b \leq K + 4$.
- ▹ The points $(a, b) \in G_M$ such that $N - 2 \leq a \leq N$ and $N + 2 \leq b \leq K + 4$.

(b) Function Values. We define the following function

$$h_{LB}(x, y) = -x - y - 3 \cdot M \tag{12}$$

and for every $(a, b) \in \mathcal{L}_B$ we set $g(a, b) = h_{LB}(a, b)$. As shown in Fig. 2b the function values inside the light blue region decrease as both of the coordinates x

and y increase. Another property of light blue region, that will be clear once we define all the regions, is that it has the value greater than the dark blue region but smaller than both of the red regions and the background as shown in Fig. 3.

(c) Gradient Values. The gradient values in the light blue region is either $(-\delta, 0)$ or $(0, -\delta)$. As we explained in (a) the light blue region contains two vertical segments and one horizontal segment each of which has length 2. For the vertical segments the left part has gradient $(-\delta, 0)$ and the right $(0, -\delta)$. For the horizontal segment the top part has gradient $(-\delta, 0)$ and the bottom $(0, -\delta)$. More precisely we have that:

- For the points $(a, b) \in \mathcal{L}_B$ it holds that $\nabla g(a, b) = (-\delta, 0)$ in the following cases:
  - $a = K - 3/2$ and $K + 3 \leq v \leq M - 2$,
  - $N - 2 \leq a \leq K - 1$ and $b = K + 7/2$,
  - $a = N - 3/2$ and $N + 2 \leq b \leq K + 4$.
- For the points $(a, b) \in \mathcal{L}_B$ it holds that $\nabla g(a, b) = (0, -\delta)$ in the following cases:
  - $a = K - 1/2$ and $K + 2 \leq b \leq M$,
  - $N - 1 \leq a \leq K$ and $b = K + 5/2$,
  - $a = N - 1/2$ and $N + 2 \leq b \leq K + 1$.

In Fig. 2c we show the direction of the negative gradient for all the regions.

Light Red Region. We denote this region by $\mathcal{L}_R$ and has the following specifications:

(a) Location. As shown in Fig. 2a, the light red region contains the following three segments of points, each of which as we can see as width 2, two of these are vertical segments and one is horizontal. In particular, $\mathcal{L}_R$ contains the following points:

- The points $(a, b) \in G_M$ such that $K \leq a \leq K + 2$ and $K \leq b \leq M$.
- The points $(a, b) \in G_M$ such that $N \leq a \leq K + 2$ and $K \leq b \leq K + 2$.
- The points $(a, b) \in G_M$ such that $N \leq a \leq N + 2$ and $N + 2 \leq b \leq K + 2$.

(b) Function Values. We define the following function

$$h_{LR}(x, y) = -x - y + 6 \cdot M \tag{13}$$

and for every $(a, b) \in \mathcal{L}_R$ we set $g(a, b) = h_{LR}(a, b)$. As shown in Fig. 2b the function values inside the light red region decrease as both of the coordinates x and y increase. Another property of light red region, that will be clear once we define all the regions, is that it has the value greater than all the other regions as shown in Fig. 3.

(c) Gradient Values. The gradient values in the light red region is either $(-\delta, 0)$ or $(0, -\delta)$. As we explained in (a) the light red region contains two vertical segments and one horizontal segment each of which has length 2. For the vertical segments the left part has gradient $(0, -\delta)$ and the right $(-\delta, 0)$. For the horizontal segment the top part has gradient $(0, -\delta)$ and the bottom $(-\delta, 0)$. More precisely we have that:

- For the points $(a, b) \in \mathcal{L}_R$ it holds that $\nabla g(a, b) = (-\delta, 0)$ in the following cases:
  - $a = K + 3/2$ and $K \leq b \leq M$,
  - $N + 1 \leq a \leq K + 2$ and $b = K + 1/2$,
  - $a = N + 3/2$ and $N + 2 \leq b \leq K + 1$.
- For the points $(a, b) \in \mathcal{L}_B$ it holds that $\nabla g(a, b) = (0, -\delta)$ in the following cases:
  - $a = K + 1/2$ and $K + 1 \leq b \leq M$,
  - $N \leq a \leq K + 1$ and $b = K + 3/2$,
  - $a = N + 1/2$ and $N + 2 \leq b \leq K + 2$.

In Fig. 2c we show the direction of the negative gradient for all the regions.

Background. We denote this region by $\mathcal{B}$ and has the following specifications:

(a) Location. Background is shown in Fig. 2a with white and as we can see it contains all the points that are not in any other region.

(b) Function Values. We define the following function

$$h_B(x, y) = -x + \mathbf{1}\{x \geq K\} + M \tag{14}$$

and for every $(a, b) \in \mathcal{B}$ we set $g(a, b) = h_B(a, b)$. As shown in Fig. 2b the function values inside the background decrease as the x coordinate increases and it is independent from the coordinate y. Another property of the background is that it has the value greater than the blue regions and smaller than the red regions as shown in Fig. 3.

(c) Gradient Values. The gradient values in the background is always $(-\delta, 0)$. In Fig. 2c we show the direction of the negative gradient for all the regions.

Top blue region. We denote this region by $\mathcal{T}_B$ and has the following specifications:

(a) Location. The top region consists of the two top rows of G_M and two of the middle columns, i.e., $(a, b) \in \mathcal{T}_B$ if and only if $K - 2 \leq a \leq K$ and $M - 1 \leq b \leq M$.

(b) Function Values. We define the following function

$$h_{TB}(x, y) = -x - (y - M - 1) - 6 \cdot M \tag{15}$$

and for every $(a, b) \in \mathcal{B}$ we set $g(a, b) = h_{TB}(a, b)$. As shown in Fig. 2b the function values inside the background decrease as the x coordinate increases and it is independent from the coordinate y. We treat this region as a dark blue region because they have a pretty minor difference in the values and for the purpose of all the proofs the behave exactly the same.

(c) Gradient Values. The same as in the dark blue region: for $(a, b) \in \mathcal{T}_B$, with $a = K - 1/2$ we have $\nabla g(a, b) = (0, -\delta)$ and for $(a, b) \in \mathcal{T}_B$, with $a = K - 3/2$ we have $\nabla g(a, b) = (-\delta, 0)$.

Top red region. We denote this region by $\mathcal{T}_R$ and has the following specifications:

(a) Location. The top region consists of the two top rows of G_M and two of the middle columns, i.e., $(a, b) \in \mathcal{T}_R$ if and only if $K \leq a \leq K + 2$ and $M - 1 \leq b \leq M$.

(b) Function Values. We define the following function

$$h_{TR}(x, y) = -x - (y - M - 1) + 3 \cdot M \tag{16}$$

and for every $(a, b) \in \mathcal{B}$ we set $g(a, b) = h_{TR}(a, b)$. As shown in Fig. 2b the function values inside the background decrease as the x coordinate increases and it is independent from the coordinate y. We treat this region as a dark red region because they have a pretty minor difference in the values and for the purpose of all the proofs the behave exactly the same.

(c) Gradient Values. The same as in the dark blue region: for $(a, b) \in \mathcal{T}_R$, with $a = K + 1/2$ we have $\nabla g(a, b) = (0, -\delta)$ and for $(a, b) \in \mathcal{T}_R$, with $a = K + 3/2$ we have $\nabla g(a, b) = (-\delta, 0)$.

PLS Boxes. We defer the discussion about PLS boxes for Sect. 3.2.5.

Now that we defined all the regions except from the PLS boxes we are ready to prove the following lemma.

Lemma 3 *After interpolating using the techniques discussed in Sect. 3.2.3, there is no* 0.01*-stationary point outside the region of PLS Box (A) or PLS Box (B).*

Proof In order to prove Lemma 3, we will show that any small box that does not lie in a solution region, does not contain any ε-stationary point. The behaviour of the function in a given small box depends on the information we have about the four corners, namely the colours and arrows at the four corners, but also on the position of the box in our instance, since the value defined by a colour depends on the position. As in the proof of Lemma 4.3 of [11], for our proof, it is convenient to consider a box with the (colour and arrow) information about its four corners, but without any information about its position. Indeed, if we can show that a box does not contain any ε-stationary point using only this information, then this will always hold, wherever the box is positioned. As a result, we obtain a finite number of boxes (with colour and arrow information) that we need to check. Conceptually, this is a straightforward task: for each small box we get a set of cubic polynomials that could be generated by bicubic interpolation for that box, and we must prove that no polynomial in that set has an ε-stationary point within the box. Unfortunately there is a big number of boxes that we need to check and for this reason it is convenient to cluster the possible instances that arise in four groups each of which is guaranteed not to have a solution. Every small box can be classified in one of the groups if after a combination of the following transformations we can produce exactly the directions of the arrows at the four corners of the small box that appear in the characteristic image of the group. The possible transformations are:

- ▷ *Reflection with respect to the y-axis.* Applying this transformation to a box has the following effect: the two corners at the top of the box now find themselves at the bottom of the box (and vice-versa) and the sign of the y-coordinate of each arrow is flipped. Using Eqs. (8) and (9) one can check that taking the bicubic

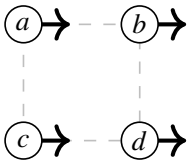
interpolation of this reflected box yields the same result as taking the interpolation of the original box and then applying the reflection to the interpolated function. We can summarize this by saying that bicubic interpolation *commutes* with this transformation.

- ▷ *Reflection with respect to the x-axis.* Similarly with the above reflection of the y-axis.
- ▷ *Reflection with respect to the axis $y = x$*, i.e., the diagonal through the box. This corresponds to swapping the corners $(0, 1)$ and $(1, 0)$ of the box, and additionally also swapping the x- and y-coordinate of the arrows at all four corners. Again, using Eqs. (8) and (9) one can directly verify that this transformation also commutes with bicubic interpolation (where applying the transformation to the interpolated function f corresponds to considering $(x, y) \mapsto f(y, x)$).
- ▷ *Reflection with respect to the axis $y = -x$.* Similarly with the above reflection with respect to the $y = -x$ axis.
- ▷ *Negation.* This corresponds to negating the values and arrows at the four corners, where "negating an arrow" just means replacing it by an arrow in the opposite direction. Using Eqs. (8) and (9), it is immediately clear that negation commutes with bicubic interpolation.

Since all five transformations commute with bicubic interpolation, this continues to hold for any more involved transformation that is constructed from these basic five. Furthermore, it is easy to see that the basic transformations do not introduce ε-stationary points. Indeed, if a function does not have any ε-stationary points, then applying any reflection or taking the negation cannot change that property. As a result, if two boxes are "symmetric" (i.e., one can be obtained from the other by applying a combination of the three basic transformations), then it is enough to verify that just one of these two boxes does not contain any ε-stationary points when we take the bicubic interpolation.

Using this notion of symmetry, the boxes that need to be verified can be grouped into just four different groups, namely Groups 1 to 4 that we define below. These are the same four groups used in [11], where it was already shown that they do not contain any 0.01-stationary points.

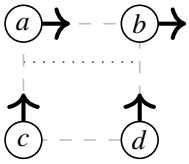
Group 1 This group contains all the boxes that are symmetric to a box of the following form:



Conditions:
$a \geq b + 1$
$c \geq d + 1$

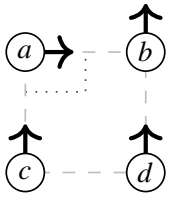
where a, b, c, d are the values at the four corners of the box as shown.

Group 2 This group contains all the boxes that are symmetric to a box of the following form:



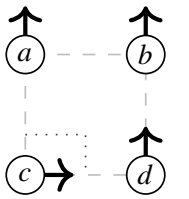
Conditions:
$a \geq b + 1$
$c \geq a + 1$
$d \geq b + 1$
$c \geq d - 1$

Group 3 This group contains all the boxes that are symmetric to a box of the following form:



Conditions:
$a \geq b + 1$
$c \geq a + 1$
$d \geq b + 1$
$c \geq d - 1$

Group 4 This group contains all the boxes that are symmetric to a box of the following form:



Conditions:
$c \geq a + 1$
$d \geq b + 1$
$c \geq d + 1$
$a \geq b - 1$

Since only the directions of the gradients and the colors of the corners of the boxes are important there is only a finite number of small cells that we need to check. In particular, as we show in Fig. 4, in the interior of $[0, 1]^2$ there are only four places where different combinations of colors and directions of gradient arise:

A. Space that only invoke the background region.
B. The left turn of the regions with light color.
C. The left turn of the regions with dark color.
D. The right turns of both of the regions that also touch in this space.
E. The place where the light and dark colors meet after we periodically repeat the construction of $[0, M]^2$.

Apart from the interior we also need to make sure that no solutions arise in the boundary after we periodically repeat the $[0, M]^2$. For this observe that both the line $x = 0$ and the line $x = M$ belong to the background and for the background function h_B it is easy to see that $h_B(0, y) = 0 = h_B(M, y)$. For the other direction we observe that the top and bottom rows have either the background, which does not depend on y and hence obviously $h_B(x, 0) = h_B(x, M)$, or it consist of the dark blue and dark red regions and the top blue and top red regions. From the definition of the top blue and the top red regions we then get that $g(x, 0) = g(x, M)$ for all $x \in [0, M]$. Now it is easy to see that using the same argument we can show that $\nabla g(a, 0) = \nabla g(a, M)$ and $\nabla g(0, b) = \nabla g(M, b)$ for all $(a, b) \in G_M$. To extend this property to all the real numbers $x, y \in [0, M]$ we observe that when we use bi-cubic interpolation the

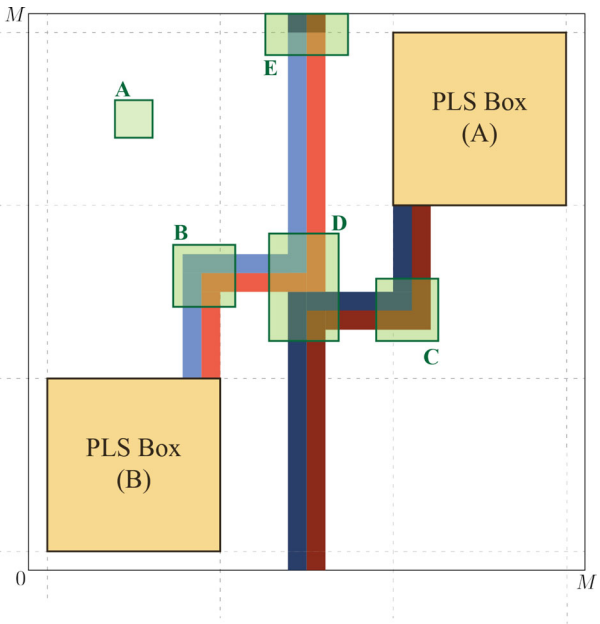


Fig. 4 In this figure we indicate with green boxes the regions that we need to check to make sure that no stationary points are created. If we make sure that there are no stationary points in these regions then we can conclude that there are no stationary points anywhere outside the PLS boxes since all the possible patterns appear in these green boxes

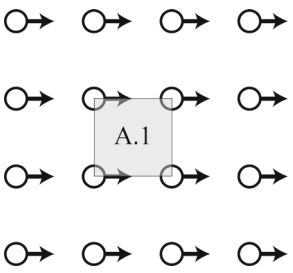


Fig. 5 Small boxes of region A of Fig. 4

values of the gradient on the edge of a small box, depends only on the specified gradients of the corners of that corresponding edge. Using this we can conclude that $\nabla g(x, 0) = \nabla g(x, M)$ and $\nabla g(0, y) = \nabla g(M, y)$ for all $x, y \in [0, 1]$. Putting all these together we conclude that the boundary conditions (5) and (6) are satisfied by our construction. We state this in the following lemma.

Lemma 4 *The boundary conditions* (5) *and* (6) *are satisfied by our construction.*

Next we verify using Groups 1. - 4. that there is no 0.01-stationary point in any of the places A, B, C, D, E and this implies that there is no 0.01-stationary point in any region outside the PLS Boxes which implies our Lemma 3.

A. We start with a figure of the region A in Fig. 5.

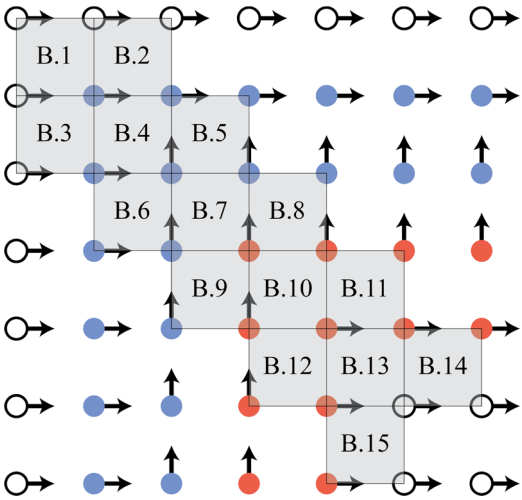


Fig. 6 Small boxes of region B of Fig. 4

As expected in this region A there is only one type of box that appears and it is of Group 1. It is easy to check that since the function value in the background decreases linearly with x, the conditions of Group 1 are satisfied and hence there is no solution in this region.

B. We start with a figure of the region B in Fig. 6, where we indicate all the small boxes with colors and gradients that have not appeared in A.

We have the following categories:

- The small boxes: B.1, B.2, B.3, B.13, B.14, B.15 follow from plain application of Group 1 using that: (1) all the colors decrease linearly as the x coordinate increases, (2) light blue is everywhere at least $\frac{1}{M}$ smaller than the background, and (3) light red is everywhere at least $\frac{1}{M}$ larger than the background.
- The small boxes: B.7, B.8, B.9 follow from application of Group 1 after taking a reflection with respect to the $y = x$ axis using that: (1) light blue and light red decrease with linear rate as the y coordinate increases, and (2) light red is everywhere at least $\frac{1}{M}$ larger than light blue.
- The small box B.4 follows from Group 3 after applying a $y = x$ reflection.
- The small box B.5 follows from a plain application of Group 2.
- The small box B.6 follows from Group 2 after applying a $y = x$ reflection.
- The small box B.10 follows from Group 3 after applying the following transformations: (i) reflection with respect to y, (ii) reflection with respect to x, and (iii) negation.
- The small box B.11 follows from Group 2 after applying the following transformations: (i) reflection with respect to y, (ii) reflection with respect to x, and (iii) negation.
- The small box B.12 follows from Group 2 after applying a $y = -x$ reflection and negation.

So no solution appears in B as well.

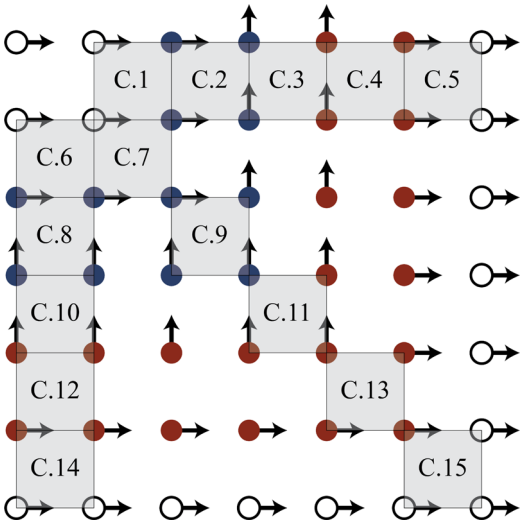


Fig. 7 Small boxes of region C of Fig. 4

C. We start with a figure of the region C in Fig. 7, where we indicate all the small boxes with colors and gradients that have not appeared in A or B.

We have the following categories:

- The small boxes: C.1, C.5, C.6, C.7, C.14, C.15 follow from plain application of Group 1 using that: (1) all the colors decrease linearly as the x coordinate increases, (2) dark blue is everywhere at least $\frac{1}{M}$ smaller than the background, and (3) dark red is everywhere at least $\frac{1}{M}$ larger than the background.
- The small boxes: C.3, C.10, C.11 follow from application of Group 1 after taking a reflection with respect to the $y = x$ axis using that: (1) dark blue and dark red decrease with linear rate as the y coordinate increases, and (2) dark red is everywhere at least $\frac{1}{M}$ larger than dark blue.
- The small box C.2 follows from Group 2 after applying a $y = x$ reflection.
- The small box C.4 follows from Group 2 after applying a $y = -x$ reflection and negation.
- The small box C.8 follows from a plain application of Group 2.
- The small box C.9 follows from a plain application of Group 3.
- The small box C.12 follows from Group 3 after applying the following transformations: (i) reflection with respect to y, (ii) reflection with respect to x, and (iii) negation.
- The small box C.13 follows from Group 3 after applying a $y = -x$ reflection and negation.

So no solution appears in C as well.

D. We start with a figure of the region D in Fig. 8, where we indicate all the small boxes with colors and gradients that have not appeared in A, B, or C.

- The small boxes: D.1, D.5, D.6, D.7, D.11 follow from plain application of Group 1 using that: (1) all the colors decrease linearly as the x coordinate increases, (2) light and dark blue are everywhere at least $\frac{1}{M}$ smaller than the background, (3)

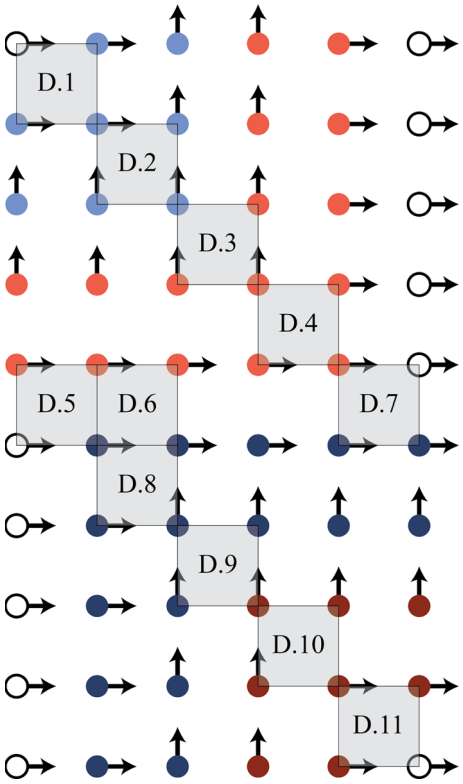


Fig. 8 Small boxes of region D of Fig. 4

light and dark red are everywhere at least $\frac{1}{M}$ larger than the background, and (4) light and dark red are everywhere at least $\frac{1}{M}$ larger than light or dark blue.

- ► The small boxes: D.3, D.9 follow from application of Group 1 after taking a reflection with respect to the $y = x$ axis using that: (1) all blue and red colors decrease with linear rate as the y coordinate increases, and (2) light and dark red are everywhere at least $\frac{1}{M}$ larger than light or dark blue.
- ► The small box D.2 follows from a plain application of Group 3.
- ► The small box D.4 follows from Group 3 after applying a $y = -x$ reflection and negation.
- ► The small box D.8 follows from Group 3 after applying a $y = x$ reflection.
- ► The small box D.10 follows from Group 3 after applying the following transformations: (i) reflection with respect to y, (ii) reflection with respect to x, and (iii) negation.

So no solution appears in D as well.

E. We start with a figure of the region E in Fig. 9, where we indicate all the small boxes with colors and gradients that have not appeared in A, B, C, or D.

- ► The small boxes: E.1, E.5 follow from plain application of Group 1 using that: (1) all the colors decrease linearly as the x coordinate increases, (2) light and dark

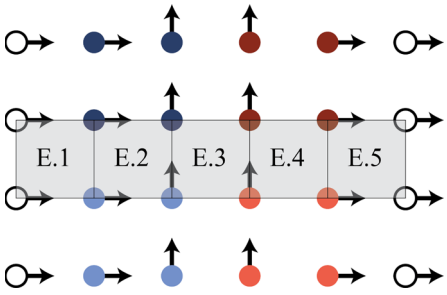


Fig. 9 Small boxes of region E of Fig. 4

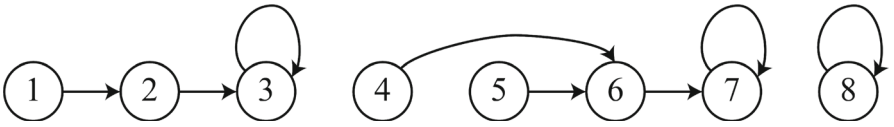


Fig. 10 Instance of the ITER problem. The nodes correspond to the nodes of the set $[2^n]$ for $n = 3$ and the arrows correspond to the output of the circuit C. The solutions are the nodes 2 and 6 in this example

blue are everywhere at least $\frac{1}{M}$ smaller than the background, and (3) light and dark red are everywhere at least $\frac{1}{M}$ larger than the background.

- The small box E.3 follows from application of Group 1 after taking a reflection with respect to the $y = x$ axis using that: (1) light blue is at least $\frac{1}{M}$ larger than dark blue, (2) light read is at least $\frac{1}{M}$ larger than dark red.
- The small box E.2 follows from Group 2 after applying a $y = x$ reflection and using the fact that light blue is larger than dark blue.
- The small box E.4 follows from Group 2 after applying a $y = -x$ reflection and negation and using the fact that light red is larger than dark red.

So no solution appears in E as well.

Since no solutions have appeared in any of A, B, C, D, E and there are no other types of small boxes that appear outside the PLS boxes we conclude that Lemma 3 follows. □

3.2.5 PLS boxes

We are now ready to define the PLS boxes. The construction of the PLS boxes follows the high level idea of the PLS Labyrinth of [11] but adapted to fit the periodic construction that we described above. We start with an instance of the ITER problem and we want to encode it inside both of the PLS Box (A) and the PLS Box (B). For the illustration of the construction we use the ITER instance that we show in Fig. 10.

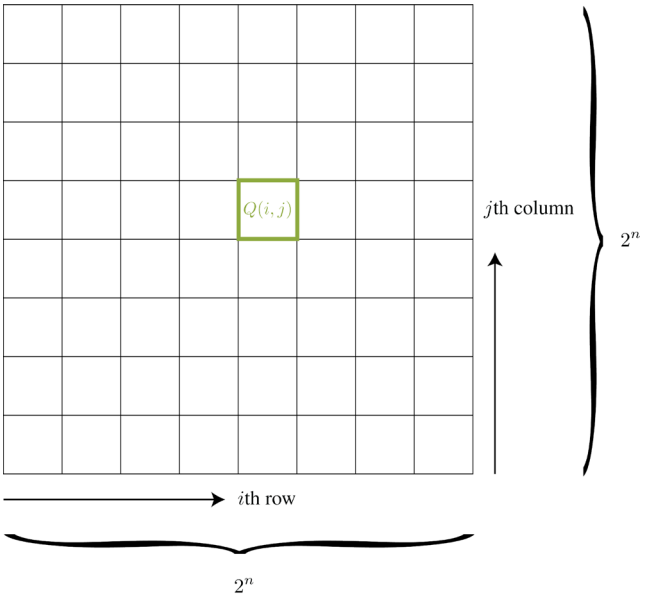
We begin with the description of the PLS Box (A) and then we describe PLS Box (B) which is a symmetric version of (A). Before that we give some general definitions that are needed for both of the boxes. Let assume that we are given an instance of ITER, i.e., a boolean circuit $C : [2^n] \to [2^n]$ for some $n \in \mathbb{N}$. The PLS Box contains a subgrid of size $N \times N$. We split this subgrid into $2^n \times 2^n$ *medium boxes* of size 8×8. Hence, we pick $N = 2^{n+3}$. For simplicity we index all the rows and columns of the

PLS boxes starting from 0 to N ignoring the constant offset of the placement of the PLS boxes inside the grid G_M.

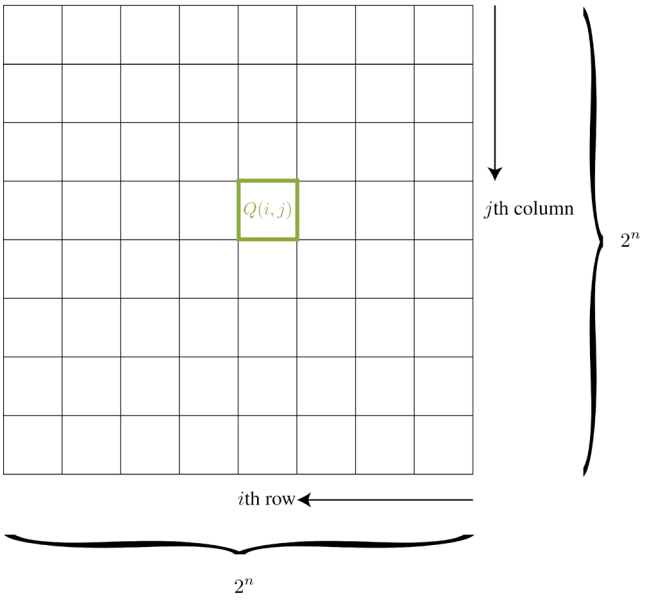
PLS Box (A). In Fig. 11(a) we show how we split the $N \times N$ subgrid of G_M into $2^n \times 2^n$ medium boxes, and we also show that we use $Q(i, j)$ to indicate the medium box that is in the ith row of medium boxes and the jth column of medium boxes.

(a) Regions. The locations of the regions for constructing the PLS Box (A) are the following

- **Horizontal Dark Red Line.** We recall the definition of the regions outside of the PLS boxes. In particular, as shown in Fig. 2a, on the bottom left corner of the PLS Box (A) there is a dark blue & dark red line that goes inside the PLS box. The first thing in the construction of the PLS box is to extend a horizontal line, in the bottom two rows of the PLS Box, with dark red color starting from the bottom of the medium box $Q(1, 1)$ all the way until the box $Q(2^n, 1)$ as shown in the example of Fig. 12.
- **Initial Node.** The initial node corresponds to the node 1 of the ITER instance. For example see Fig. 10. The implementation of this initial node is to continue vertically the dark blue & dark red line that touches the PLS Box (A) from outside for 4 rows up.
- **Nodes $u \in [2^n]$ with $C(u) > u$.** These are the nodes of ITER that do not have a self-loop and are potential solutions to ITER depending on whether $C(C(u)) > C(u)$ or not. Independently of whether u is a solution, for every u such that $C(u) > u$ we start from the $Q(u, 1)$ medium box until the $Q(u, u)$ box a vertical dark blue & dark red line where each of the colors have width 2. The 4 columns that this line uses are the 4 columns in the middle of $Q(u, 1)$, i.e., the 4 small columns in the middle of the uth column of medium boxes. Inside $Q(u, 1)$ this vertical dark blue & dark red line starts from the third row of $Q(u, 1)$, i.e., right above the horizontal dark red line that we described above. Inside $Q(u, u)$ the vertical dark blue & dark red line stops at the fourth row of $Q(u, u)$, i.e., in the middle of the medium box $Q(u, u)$.
- **Connection of $u \in [2^n]$ with $C(u)$ if $C(u) > u$ and $C(C(u)) > C(u)$.** If $C(u) > u$ then already as we described construct a vertical dark blue & dark red line that ends in the middle of $Q(u, u)$. Now if we additionally have that $C(C(u)) > C(u)$ then in the middle of $Q(u, u)$ right above the end of the vertical dark blue & dark red line we start a horizontal dark blue line with width 2 that continues until the medium box $Q(C(u), u)$. Since $C(C(u)) > C(u)$ and $C(u) > u$ this means that according to the rule above there will be a vertical dark blue & dark red line that goes through the box $Q(C(u), u)$. This dark blue & dark red line will have the blue to the left and the red the right. Our horizontal dark blue line will stop once it hits the vertical dark blue line in $Q(C(u), u)$.
- **Crossing between horizontal dark blue lines and vertical dark blue & dark red lines.** Depending on the instance, it could be that there exist two vertices $u, v \in [2^n]$ such that: $C(u) > u$, $C(C(u)) > C(u)$, $u < v < C(u)$, and $C(v) > v$. For example this is true in the example of Fig. 10 with $u = 4$ and $v = 5$. In that case, in the box $Q(v, u)$ the horizontal dark blue line from $Q(u, u)$ to $Q(C(u), u)$ and the vertical dark blue & dark red line from $Q(v, 1)$ to $Q(v, v)$ cross. When



(a) PLS Box (A).



(b) PLS Box (B).

Fig. 11 The medium boxes and their indexing rules in the PLS Box (A) and the PLS Box (B). The arrows indicate the direction of increase of i and j in PLS Box (A) and the PLS Box (B) respectively

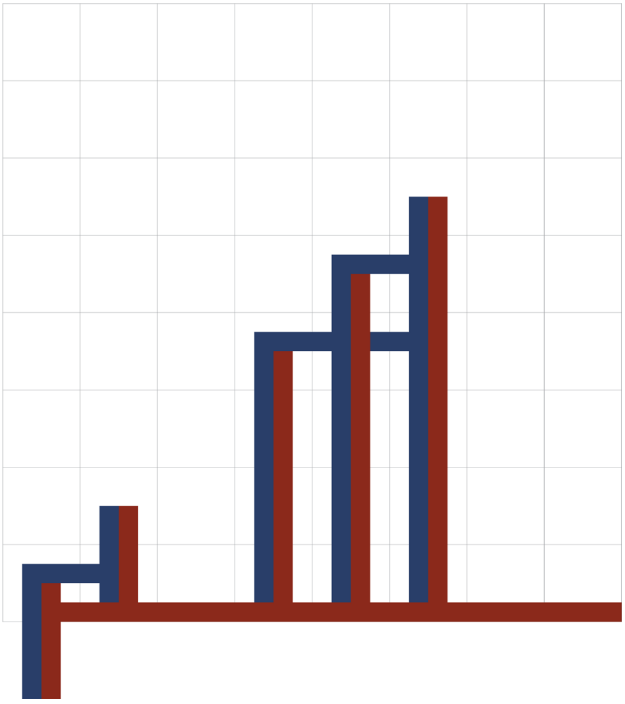


Fig. 12 The regions of PLS Box (A) when implementing the ITER instance provided in Fig. 10

this happens then inside $Q(v, u)$ the vertical dark blue & dark red line overwrites the horizontal dark blue line and otherwise the lines occupy the small boxes that they were supposed to.

- **Background.** Every point in the grid that does not belong to any of the aforementioned regions belongs to the background which is defined the same way that we defined the background in Sect. 3.2.4.

We illustrate the above construction for the ITER example of Fig. 10 in Fig. 12.

(b) Function Values. The function values are defined exactly the same as we described in Sect. 3.2.4 for the regions: dark blue, dark red, and background. In particular, using the functions h_B, h_{DB}, h_{DR}.

(c) Gradient Values. The gradient values follow the following rules:

- **Horizontal dark red lines.** The gradients in the horizontal dark red lines are always $(-\delta, 0)$ expect from the following places: for any $u \in [2^n]$ such that $C(u) > u$, in the middle of the medium box $Q(u, 1)$ in the row before the vertical dark blue & dark red line and directly below the right dark blue line and the left dark red line in these two grid points the gradient is $(0, -\delta)$.
- **Vertical dark blue & dark red lines.** In this line the left dark blue lines has gradients $(-\delta, 0)$, the right dark blue line has gradients $(0, -\delta)$, the left dark red line has gradients $(0, -\delta)$ and the right dark red line has gradients $(-\delta, 0)$.

- **Horizontal dark blue lines.** The gradients in the horizontal dark blue lines are always $(-\delta, 0)$ expect from the following places: for any $u \in [2^n]$ such that $C(u) > u$ and $C(C(u)) > C(u)$ the, in the middle of the medium box $Q(u, u)$ in the row above the vertical dark blue & dark red line and directly above the right dark blue line and the left dark red line in these two grid points the gradient is $(0, -\delta)$.
- **Background.** The gradient in the grid points of the background is always $(-\delta, 0)$.

The rest of the function in PLS Box (A) is defined via the bicubic interpolation that we described in Sect. 3.2.3. This completes the description of the PLS Box (A).

PLS Box (B). We again split the $N \times N$ subgrid of G_M into $2^n \times 2^n$ medium boxes, and we use $Q(i, j)$ to indicate the medium box that is in the ith row of medium boxes and the jth column of medium boxes but when instead of starting the counting from the bottom left corner, we start the counting from the top right corner, as we show in Fig. 11b.

(a) Regions. The locations of the regions for constructing the PLS Box (B) are the following

- **Horizontal Light Blue Line.** We recall the definition of the regions outside of the PLS boxes. In particular, as shown in Fig. 2a, on the top right corner of the PLS Box (B) there is a light blue & light red line that gets inside the PLS box. The first thing in the construction of the PLS box is to extend a horizontal line, in the top two rows of the PLS Box, with light blue color starting from the top of the medium box $Q(1, 1)$ all the way until the box $Q(2^n, 1)$ as shown in the example of Fig. 13.
- **Initial Node.** The initial node corresponds to the node 1 of the ITER instance. For example see Fig. 10. The implementation of this initial node is to continue vertically the light blue & light red line that touches the PLS Box (B) from outside for 4 rows down.
- **Nodes $u \in [2^n]$ with $C(u) > u$.** These are the nodes of ITER that do not have a self-loop and are potential solutions to ITER depending on whether $C(C(u)) > C(u)$ or not. Independently of whether u is a solution, for every u such that $C(u) > u$ we start from the $Q(u, 1)$ medium box until the $Q(u, u)$ box a downwards vertical light blue & light red line where each of the colors have width 2. The 4 columns that this line uses are the 4 columns in the middle of $Q(u, 1)$, i.e., the 4 small columns in the middle of the uth column of medium boxes. Inside $Q(u, 1)$ this vertical light blue & light red line starts from the third row of $Q(u, 1)$, i.e., right above the horizontal light blue line that we described above. Inside $Q(u, u)$ the vertical light blue & light red line stops at the fourth row of $Q(u, u)$, i.e., in the middle of the medium box $Q(u, u)$.
- **Connection of $u \in [2^n]$ with $C(u)$ if $C(u) > u$ and $C(C(u)) > C(u)$.** If $C(u) > u$ then already as we described construct a downwards vertical light blue & light red line that ends in the middle of $Q(u, u)$. Now if we additionally have that $C(C(u)) > C(u)$ then in the middle of $Q(u, u)$ right below the end of the vertical light blue & light red line we start a horizontal light red line with width 2 that continues until the medium box $Q(C(u), u)$. Since $C(C(u)) > C(u)$ and $C(u) > u$ this means that according to the rule above there will be a vertical light blue & light red line that goes downwards through the box $Q(C(u), u)$. This light blue

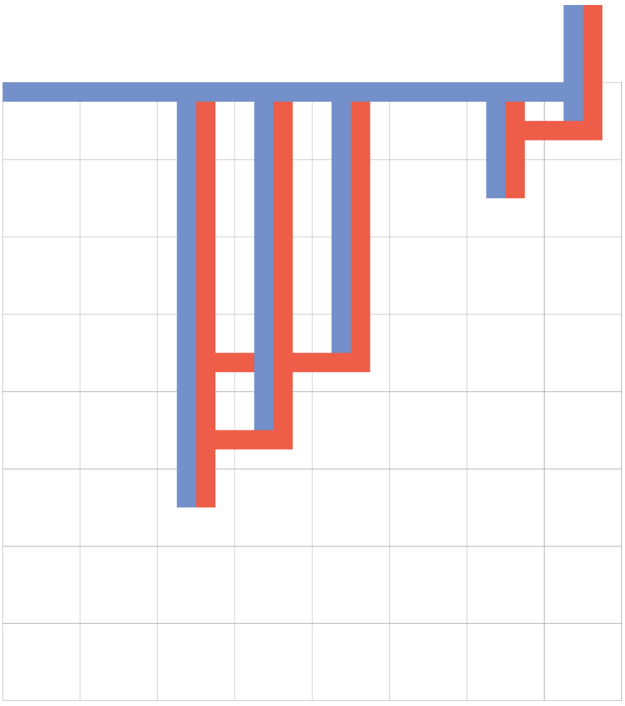


Fig. 13 The regions of PLS Box (B) when implementing the ITER instance provided in Fig. 10

& light red line will have the blue to the left and the red the right. Our horizontal light red line, which in this case is coming from the right, will stop once it hits the vertical light red line in $Q(C(u), u)$.

- **Crossing between horizontal light red lines and vertical light blue & light red lines.** Depending on the instance, it could be that there exist two vertices $u, v \in [2^n]$ such that: $C(u) > u$, $C(C(u)) > C(u)$, $u < v < C(u)$, and $C(v) > v$. For example this is true in the example of Fig. 10 with $u = 4$ and $v = 5$. In that case, in the box $Q(v, u)$ the horizontal light red line from $Q(u, u)$ to $Q(C(u), u)$ and the vertical light blue & light red line from $Q(v, 1)$ to $Q(v, v)$ cross. When this happens then inside $Q(v, u)$ the vertical light blue & light red line overwrites the horizontal light red line and otherwise the lines occupy the small boxes that they were supposed to.
- **Background.** Every point in the grid that does not belong to any of the aforementioned regions belongs to the background which is defined the same way that we defined the background in Sect. 3.2.4.

We illustrate the above construction for the ITER example of Fig. 10 in Fig. 13.

(b) Function Values. The function values are defined exactly the same as we described in Sect. 3.2.4 for the regions: light blue, light red, and background. In particular, using the functions h_B, h_{LB}, h_{LR}.

(c) Gradient Values. The gradient values follow the following rules:

- **Horizontal light blue lines.** The gradients in the horizontal light blue lines are always $(-\delta, 0)$ expect from the following places: for any $u \in [2^n]$ such that $C(u) > u$, in the middle of the medium box $Q(u, 1)$ in the row after the vertical light blue & light red line and directly above the right light blue line and the left light red line, in these two grid points, the gradient is $(0, -\delta)$.
- **Vertical light blue & light red lines.** In this line the left light blue lines has gradients $(-\delta, 0)$, the right light blue line has gradients $(0, -\delta)$, the left light red line has gradients $(0, -\delta)$ and the light dark red line has gradients $(-\delta, 0)$.
- **Horizontal light red lines.** The gradients in the horizontal light red lines are always $(-\delta, 0)$ expect from the following places: for any $u \in [2^n]$ such that $C(u) > u$ and $C(C(u)) > C(u)$ the, in the middle of the medium box $Q(u, u)$ in the row below the vertical light blue & light red line and directly below the right light blue line and the left light red line, in these two grid points, the gradient is $(0, -\delta)$.
- **Background.** The gradient in the grid points of the background is always $(-\delta, 0)$.

The rest of the function in PLS Box (B) is defined via the bicubic interpolation that we described in Sect. 3.2.3. This completes the description of the PLS Box (B).

We are now ready to prove the following lemma.

Lemma 5 *After interpolating using the techniques discussed in Sect. 3.2.3, there is no* 0.01*-stationary point in any medium box of PLS Box (A) and PLS Box (B), except medium boxes* $Q(u, u)$ *for which* $C(u) > u$ *and* $C(C(u)) = C(u)$.

Proof (of Lemma 5) For this proof we will again use the groups of small boxes that we introduced in the proof of Lemma 3.

We start with identifying the regions in PLS Box (A) and PLS Box (B) that we need to check, shown in Figs. 14 and 15 respectively. In particular we need to check the following regions in PLS Box (A):

F. The initial node in PLS Box (A) where the horizontal dark red line starts.
G. The start of a vertical dark blue & dark red line.
H. The end of a vertical dark blue & dark red line and start of a horizontal dark blue line.
I. The crossing of a horizontal dark blue line and a vertical dark blue & dark red line.
J. The end of a horizontal dark blue line.
K. The end of the horizontal dark red line.

and symmetrically the following regions in PLS Box (B):

L. The initial node in PLS Box (B) where the horizontal light blue line starts.
M. The start of a vertical light blue & light red line.
N. The end of a vertical light blue & light red line and start of a horizontal light red line.
O. The crossing of a horizontal light red line and a vertical light blue & light red line.
P. The end of a horizontal light red line.
R. The end of the horizontal light blue line.

These regions are shown in Figs. 14 and 15. We intentionally left out of the above lists the end of the vertical dark blue & dark red lines with the start of a horizontal

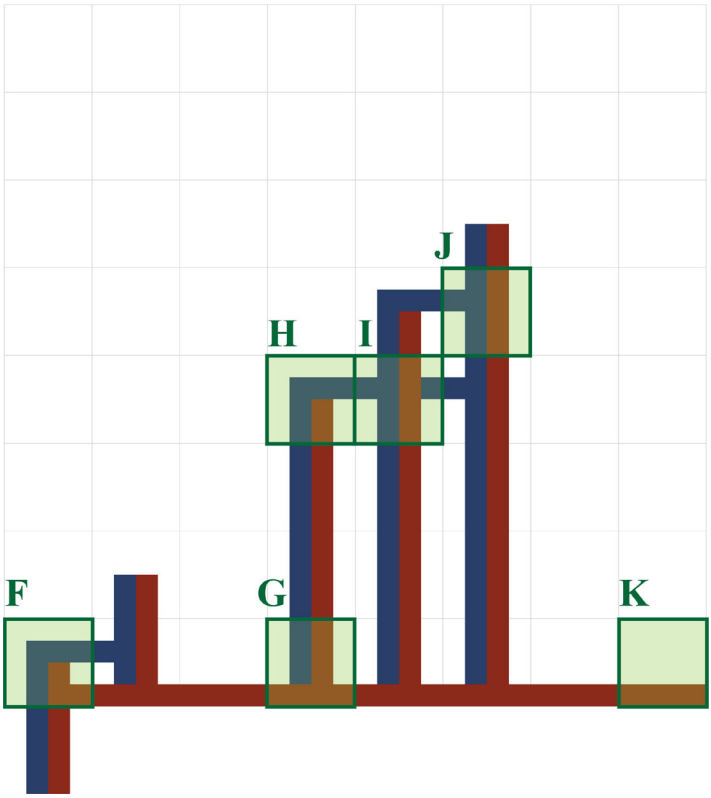


Fig. 14 With the green squares we indicate the regions, with their labels, that we need to analyze in the proof of Lemma 5 for the case of PLS Box (A)

dark blue line because these correspond exactly to medium boxes $Q(u, u)$ such that $C(u) > u$ but $C(C(u)) = C(u)$ which correspond to solutions of the ITER instance. Hence if there are solutions in these medium boxes then we recover a solution of the ITER instance which is what we want. For the same reason we left out of the above lists the end of a vertical light blue & light red path without the start of a horizontal light red path.

F. We start with a figure of the region F in Fig. 16, where we indicate all the small boxes with colors and gradients that have not appeared in A - E.

Most of the small boxes have appeared before except from the following:

- The small box F.1 follows from Group 4 after applying a $y = x$ reflection.
- The small box F.2 follows from Group 2 after applying a $y = -x$ reflection and negation.
- The small boxes F.3, F.4, F.5 follow from plain application of Group 1 using that: (1) all the colors decrease linearly as the x coordinate increases, (2) dark red is everywhere at least 1 larger than the background.

So no solution appears in F.

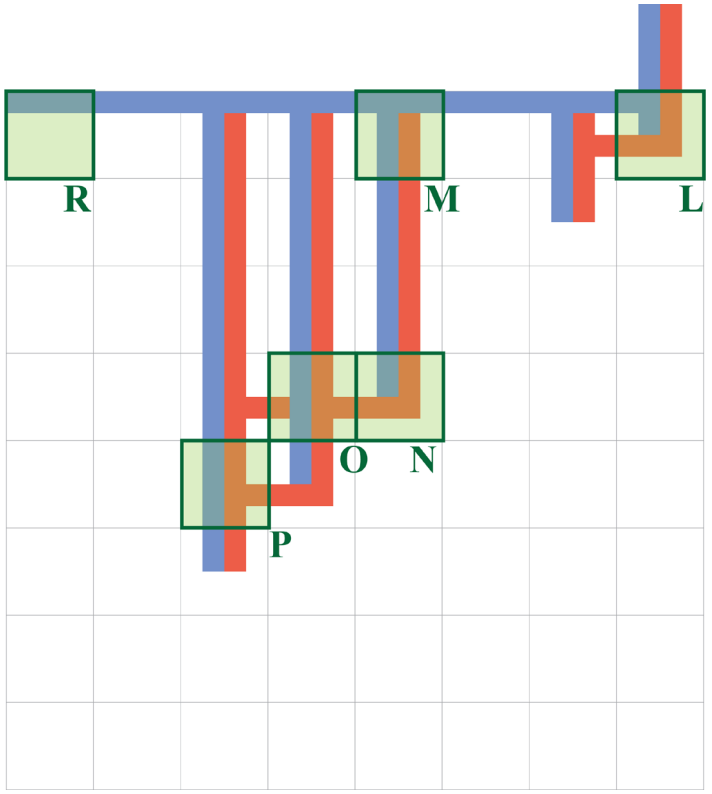


Fig. 15 With the green squares we indicate the regions, with their labels, that we need to analyze in the proof of Lemma 5 for the case of PLS Box (B)

G. We start with a figure of the region G in Fig. 17, where we indicate all the small boxes with colors and gradients that have not appeared in A – F.

Most of the small boxes have appeared before except from the following:

- The small box G.1 follows from Group 2 after applying a $y = x$ reflection.
- The small box G.2 follows from Group 4 after applying a $y = -x$ reflection and negation.

So no solution appears in G.

H. In region H, as we can see in Fig. 18, all the small boxes have appeared before in regions A–G and hence we can directly conclude that there are no solutions in H.

I. In region I, as we can see in Fig. 19, all the small boxes have appeared before in regions A–G and hence we can directly conclude that there are no solutions in I.

J. In region J, as we can see in Fig. 20, all the small boxes have appeared before in regions A–G and hence we can directly conclude that there are no solutions in J.

K. In region K, as we can see in Fig. 21, all the small boxes have appeared before in regions A–G and hence we can directly conclude that there are no solutions in K.

L. We start with a figure of the region L in Fig. 22, where we indicate all the small boxes with colors and gradients that have not appeared in A–K.

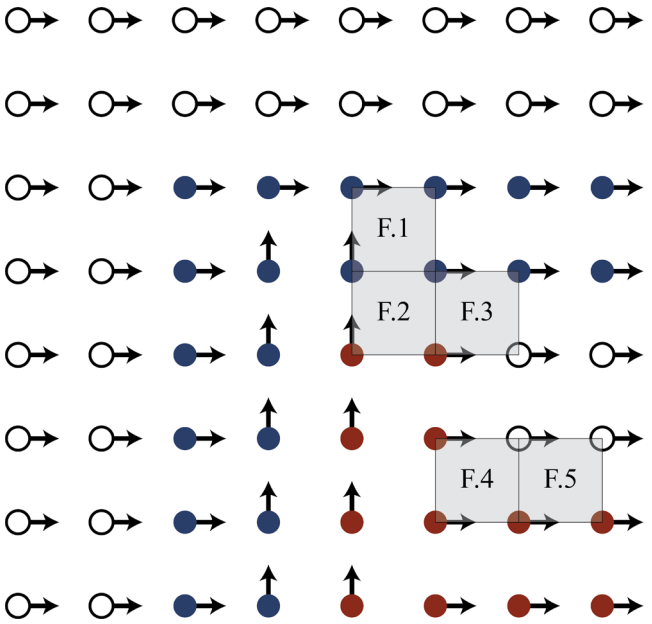


Fig. 16 The small boxes of the region F shown in Fig. 14

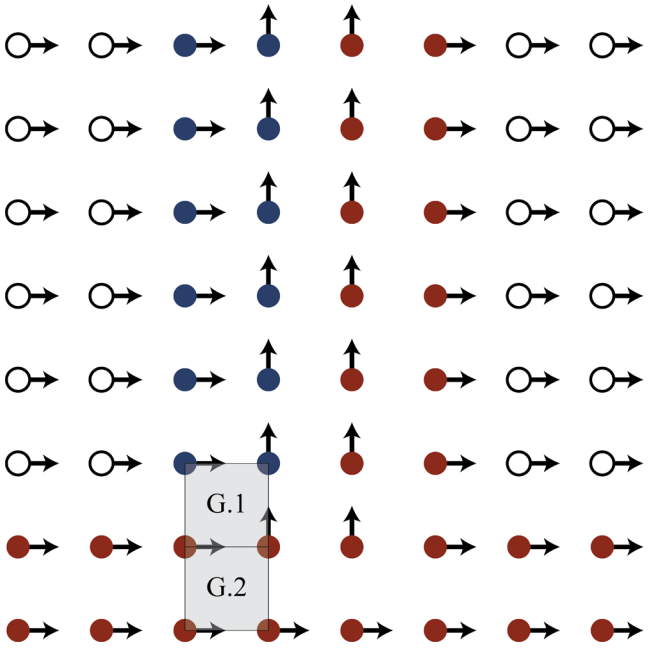


Fig. 17 The small boxes of the region G shown in Fig. 14

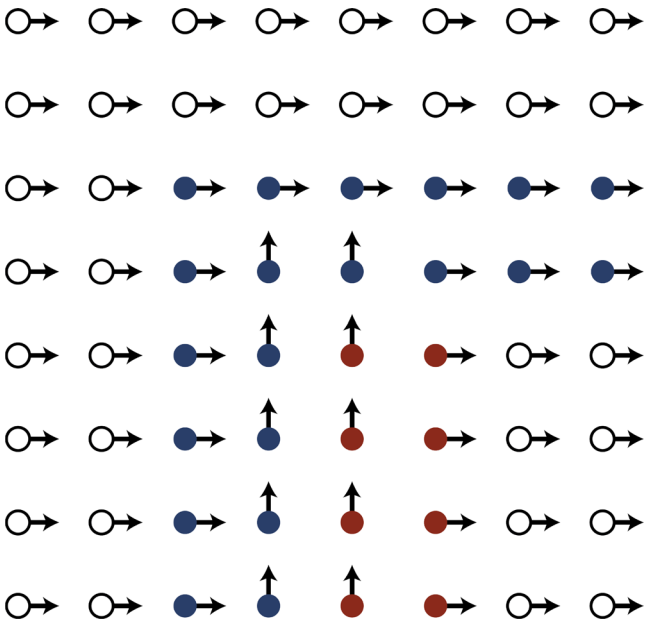


Fig. 18 The small boxes of the region H shown in Fig. 14

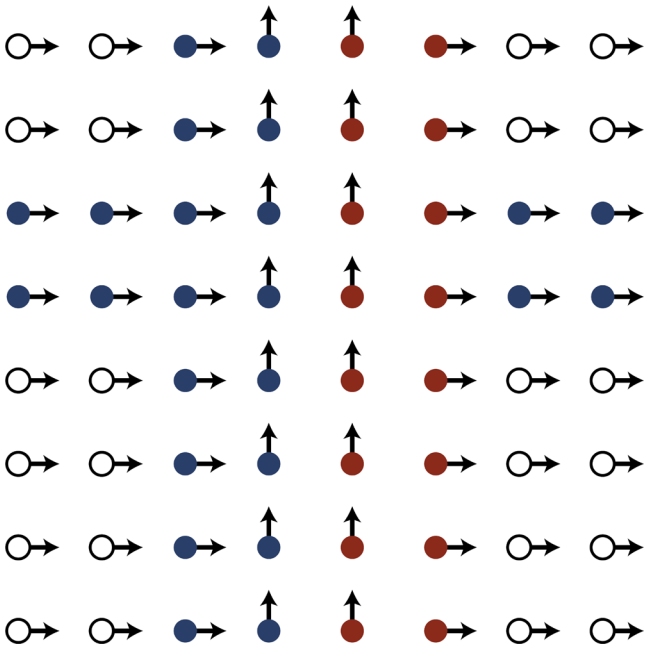


Fig. 19 The small boxes of the region I shown in Fig. 14

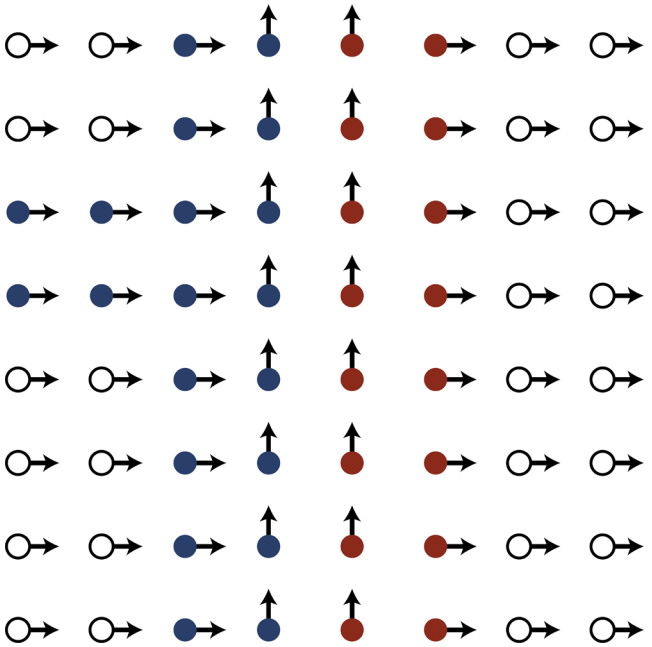


Fig. 20 The small boxes of the region J shown in Fig. 14

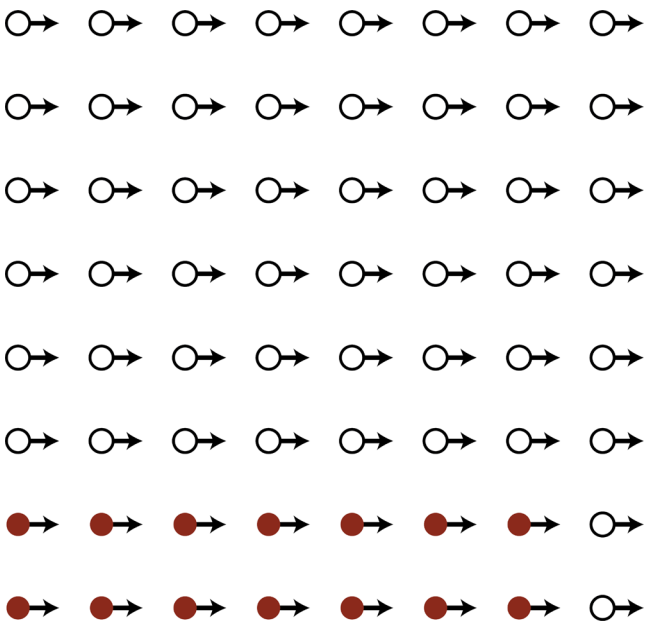


Fig. 21 The small boxes of the region K shown in Fig. 14

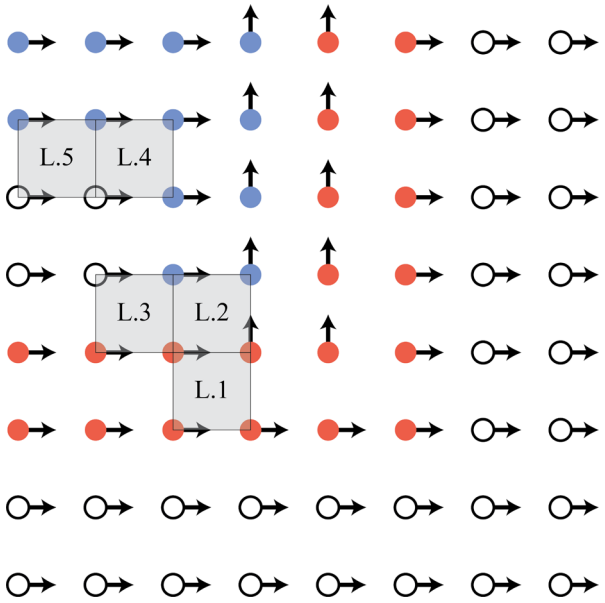


Fig. 22 The small boxes of the region L shown in Fig. 15

Most of the small boxes have appeared before except from the following:

- The small box L.1 follows from Group 4 after applying a $y = -x$ reflection and negation.
- The small box L.2 follows from Group 2 after applying a $y = x$ reflection.
- The small boxes L.3, L.4, L.5 follow from plain application of Group 1 using that: (1) all the colors decrease linearly as the x coordinate increases, and (2) light blue is everywhere at least 1 smaller than the background.

So no solution appears in L.

M. We start with a figure of the region M in Fig. 23, where we indicate all the small boxes with colors and gradients that have not appeared in A–L.

Most of the small boxes have appeared before except from the following:

- The small box M.1 follows from Group 2 after applying a $y = -x$ reflection and negation.
- The small box M.2 follows from Group 4 after applying a $y = x$ reflection.

So no solution appears in M.

N. In region N, as we can see in Fig. 24, all the small boxes have appeared before in regions A–M and hence we can directly conclude that there are no solutions in N.

O. In region O, as we can see in Fig. 25, all the small boxes have appeared before in regions A–M and hence we can directly conclude that there are no solutions in O.

P. In region P, as we can see in Fig. 26, all the small boxes have appeared before in regions A–M and hence we can directly conclude that there are no solutions in P.

R. In region R, as we can see in Fig. 27, all the small boxes have appeared before in regions A–M and hence we can directly conclude that there are no solutions in R.

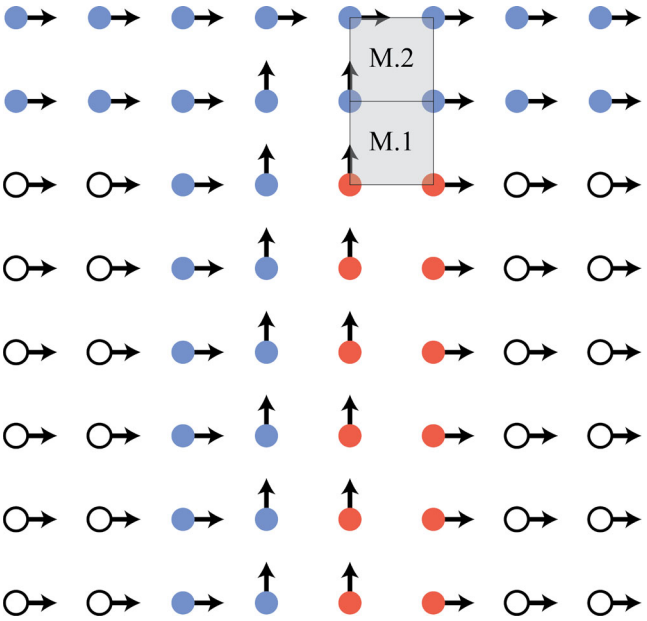


Fig. 23 The small boxes of the region M shown in Fig. 15

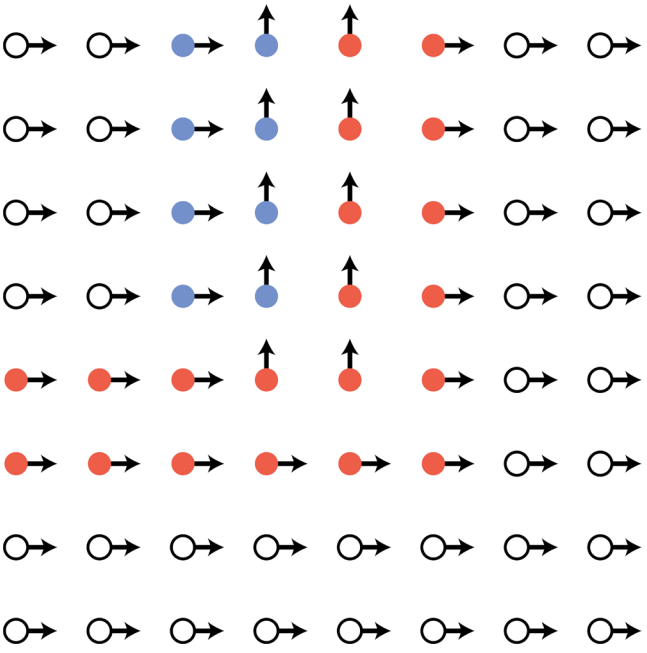


Fig. 24 The small boxes of the region N shown in Fig. 15

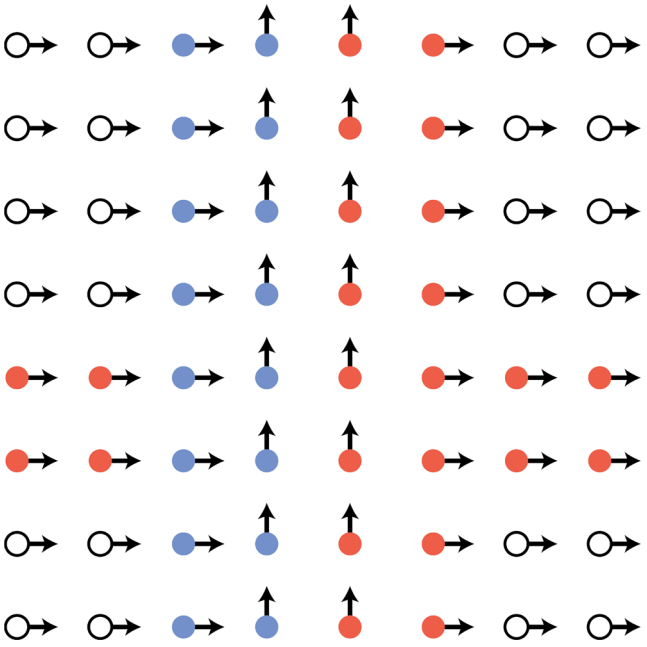


Fig. 25 The small boxes of the region O shown in Fig. 15

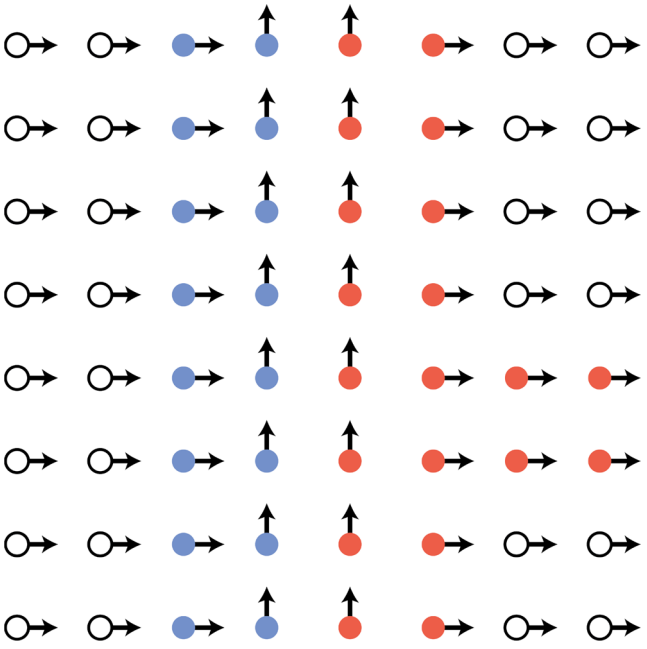


Fig. 26 The small boxes of the region P shown in Fig. 15

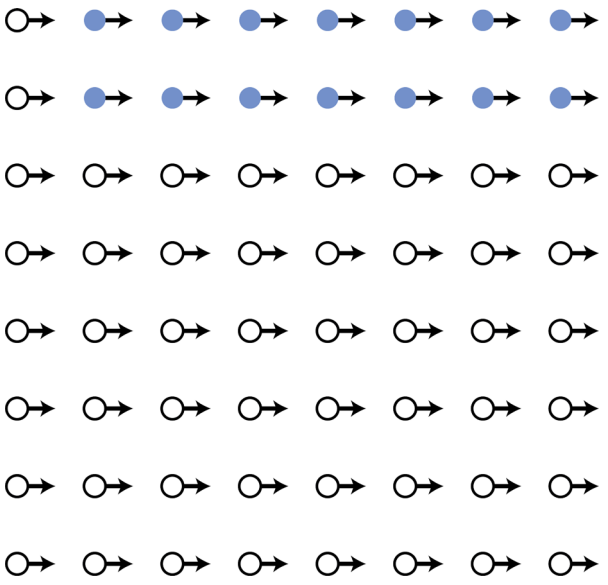


Fig. 27 The small boxes of the region R shown in Fig. 15

Since no solutions have appeared in any of F–R and there are no other types of small boxes that appear inside the PLS Box (A) and PLS Box (B), we conclude that Lemma 5 follows. □

Now combining Lemma 3 and Lemma 5 we conclude that using our construction any 0.01-stationary point can only appear in places that correspond to solutions of the original ITER instance. To finish our proof of Lemma 2 it remains: (1) to understand the boundness, Lipschitzness, and smoothness of g, and (2) to show that the complexity of computing g at every point (x, y) is polynomial in the representations of x, y and in the size of the binary circuit C from the ITER instance. We proceed with these goals in the following sections.

3.2.6 Function values, lipschitzness, and smoothness of *g*

Our goal in this section is to prove the following lemma that provides bounds on the function values, the Lipschitzness, and the smoothness of g and hence of f. The proof is essentially the same, almost verbatim, with the proof of Lemma 4.2 of [11] with some small changes to be applied in our case.

Lemma 6 *The function* $f : \mathbb{R}^2 \to \mathbb{R}$ *defined in* (4) *has the following properties:*

1. *It is continuously differentiable on* $\mathbb{R}^2$.
2. f *and its gradient* ∇f *are Lipschitz-continuous on* $\mathbb{R}^2$ *with Lipschitz-constant* $L = 2^{18}M$.
3. *It holds that* $|f(x, y)| \leq 2^{14}M$ *for all* $x, y \in \mathbb{R}^2$.

Proof The first point is a simple consequence of [17].

Next we prove the properties 2. and 3.

Lipschitz-continuity. In order to prove the second point, we first show that g and ∇g are L-Lipschitz-continuous in every small box of the grid G_M. Consider any small box. In our construction, the values of g_x, g_y at the corners of the box are upper bounded by $\delta = 1/2$. The value of g at the corners is upper bounded by the maximum value of the light red function h_{LR} which is everywhere less than $6 \cdot M$. Furthermore, the value of g at the corners is lower bounded by the minimum value of the dark blue function h_{DB} which is everywhere at least $-8 \cdot M$. So we conclude that $|g(x, y)| \leq 8\,M$ for all $(x, y) \in G_M$. Thus, using Eq. (9), it is easy to check that $|a_{ij}| \leq 2^{10} M$ for all $i, j \in \{0, 1, 2, 3\}$. Furthermore, the partial derivatives of g inside the small box can be written as:

$$\frac{\partial g}{\partial x}(x, y) = \sum_{i=1}^{3}\sum_{j=0}^{3} i \cdot a_{ij} x^{i-1} y^j \qquad \frac{\partial g}{\partial y}(x, y) = \sum_{i=0}^{3}\sum_{j=1}^{3} j \cdot a_{ij} x^i y^{j-1} \tag{17}$$

where in the above expressions $(x, y) \in [0, 1]^2$ and correspond to the local coordinates inside the small box. Finally, it is easy to see that the monomials $x^i y^j, i, j \in \{0, 1, 2, 3\}$, are 6-Lipschitz continuous over $[0, 1]^2$. Now, using Eqs. (8) and (17), we obtain that g and ∇g are Lipschitz-continuous (w.r.t. the ℓ_2-norm) with Lipschitz constant $L = 2^{18} M$ inside the small box.

Now, since g and ∇g are L-Lipschitz-continuous inside every small box and continuous over all of $[0, M]^2$, a simple argument shows that they are also L-Lipschitz-continuous over all of $[0, M]^2$ (e.g., see the proof of Lemma 4.2 in [11]). Since f is itself a tiling of g we can use the same argument together with Lemma 4 to conclude that f is also L-Lipschitz and L-smooth.

Bounded Value. As shown above we have $|g(x, y)| \leq 8\,M$ for all $(x, y) \in G_M$, and $|a_{ij}| \leq 2^{10} M$ for all $i, j \in \{0, 1, 2, 3\}$. It follows from Eq. (8) that $|g(x, y)| \leq 2^{14} M$ for all $(x, y) \in [0, M]^2$. Finally, since f is everywhere equal to the output of g at some point we get that $|f(x, y)| \leq 2^{14} M$ over the whole domain. □

3.2.7 Turing machine that evaluates *f*

The last part of the proof of Lemma 2 is to show that there exist a Turing machine $\mathcal{C}_f$ such that given two numbers $x, y \in [0, 1]$ with bit complexity $b \geq \text{len}(x)$ and $b \geq \text{len}(y)$ we can compute the value and the gradient of f at any point in time that is polynomial in b and in the size of the boolean circuit C that we are given as an input that describes the ITER instance that we are reducing from. It is easy to see that the computation $z \mapsto z - M \cdot \lfloor \frac{z}{M} \rfloor$ can be done in time polynomial in $\text{len}(z)$ and $\text{len}(M)$. $\text{len}(z)$ will be bounded by b and M is a natural number such that $M = O(2^n)$, so $\text{len}(M) = O(n)$. Since C is a circuit with n inputs and n outputs, $O(n)$ is certainly polynomial in the size of C.

Since $z \mapsto z - M \cdot \lfloor \frac{z}{M} \rfloor$ can be done in polynomial time, it suffices to show that we can compute $g(x, y)$ in polynomial time. To do that we first need to identify the type

of the small box where (x, y) belongs. To do this outside the PLS boxes, we need time linear in b. Inside the PLS boxes, on the other hand, we need to evaluate the circuit C with input u that corresponds to the column of the medium box that (x, y) belongs to, and with input v that corresponds to the row of the medium box that (x, y) belongs to. So we need to evaluate $C(u)$, $C(v)$ as well as $C(C(u))$, $C(C(v))$ and if we specify these then we can identify the type of the small box that (x, y) belongs to. So we need to evaluate C four times, which takes linear time in the size of C. Finally, once we identify the small box, we need to compute the bi-cubic interpolation which involves solving a linear system and computing a third degree polynomial with numbers that use at most $\max\{b, \text{len}(M)\}$ bits. It is well known that both of these can be done in time polynomial in the description of the number and hence we conclude that there exists an efficient Turing machine that computes g which implies an efficient Turing machine that computes f.

3.2.8 Proof of Lemma 2

To show Lemma 2 we combine Lemmas 3, 5, 4, 6 and the discussion of Sect. 3.2.7 and Lemma 2 follows.

3.3 Tight query bounds for 2D

As a corollary of the proof of Lemma 2 we can show the following black box lower bound for finding ε-stationary points in two dimensions.

Theorem 3.2 *For any deterministic algorithm $\mathcal{A}$ that computes ε-stationary points and any starting point $(x_0, y_0) \in \mathbb{R}^2$ it holds that there exists a 1-bounded, 1-Lipschitz, and 1-smooth function $f : \mathbb{R}^2 \to \mathbb{R}$, such that $\mathcal{T}(\mathcal{A}, (x_0, y_0); f) \geq \Omega\left(\frac{1}{\varepsilon}\right)$.*

Proof The first observation to show this theorem is that the black box version of ITER has a query lower bound of 2^n. Consider any algorithm $\mathcal{A}$ that solves ITER and has only query access to the input circuit C. At every step t of the algorithm $\mathcal{A}$, there exists a sequence $S_t = (u_0, u_1, \ldots, u_{i_t})$ such that $u_0 = 1$, $u_j = C(u_{j-1})$, for all $j \in [i_t]$ and the algorithm $\mathcal{A}$ has queried all the nodes $u \in S_t$. In words, S_t corresponds to the longest path that $\mathcal{A}$ has discovered start from 1. The adversary against $\mathcal{A}$ is very simple

- ▷ if the algorithm queries to learn $C(v)$ for some $v \neq u_{i+t}$ then the adversary replies $C(v) = v$,
- ▷ if the algorithm queries to learn $C(u_t)$ then the adversary replies with the smallest node that has never been queried before.

Hence, the only way for $\mathcal{A}$ find a solution at any node v is to have queried everything before v, otherwise the adversary will just continue the path S without giving a solution. The same way, the only way that $\mathcal{A}$ can find a solution at any node v is that $\mathcal{A}$ has queried everything after v as well because if there are still available nodes after v the adversary will continue the path S. This implies that any algorithm $\mathcal{A}$ has to query every node in the worst case before it finds a solution. So in the worst case any algorithm $\mathcal{A}$ will take time 2^n.

Now as we discussed in the proof of Lemma 2 the function f that we construct in Sect. 3.2 has the properties: (1) to evaluate f at any given point we need at most 4 queries to C, (2) any stationary point of f reveals a solution to ITER with input C. These two combined give us that any algorithm that finds a 0.01-stationary point in functions that we construct will in the worst-case take time at least 2^{n-2}. Now observe that the parameter M that we use to construct f in Sect. 3.2 satisfies $M = \Theta(2^n)$ which means that any algorithm that finds a 0.01-stationary point in functions that we construct will in the worst-case take time at least $\Omega(M)$. The last thing to fix is the parameters of the algorithm. Currently, the function f that we construct has boundedness $B = O(M)$, and Lipschitzness/smoothness $L = O(M)$. If we multiply the function value by $1/M$, i.e., we define $\tilde{f}(x) = \frac{1}{M} \cdot f(x)$, then to find a 0.01-stationary point for f we need to find a $\frac{1}{100\,M}$-stationary point of $\tilde{f}$ so our new target ε is $\frac{1}{100M}$. At the same time $\tilde{f}$ has now boundedness $B = O(1)$, and Lipschitzness/smoothness $L = O(1)$ and of course any algorithm would still need $\Omega(M)$ time to find a $\frac{1}{100M}$-stationary point. Hence, any algorithm that finds an ε-stationary point for $\tilde{f}$ will need in the worst case $\Omega(1/\varepsilon)$ queries and Theorem 3.2 follows. □

In the next section we present an algorithm with complexity $O(1/\varepsilon)$ which combined with Theorem 3.2 resolves the black box query complexity of finding stationary points when $d = 2$.

4 The gradient flow parallel trap algorithm

In this section we present the Gradient Flow Parallel Trap (GFPT) algorithm for computing ε-stationary points in both the unconstrained and constrained settings. This algorithm is inspired by the Gradient Flow Trapping (GFT) algorithm proposed by Bubeck and Mikulincer [4] for the constrained setting. For $d = 2$ their GFT algorithm yields an upper bound that is almost tight, namely up to $\log(1/\varepsilon)$ factors. The GFPT algorithm we propose here uses some core ideas from the GFT algorithm to achieve a *tight* upper bound for $d = 2$. Furthermore, GFPT is in fact simpler than GFT: while GFT relies on two subroutines (*parallel trap* and *edge fixing*), GFPT only uses an improved version of one of the two subroutines (namely, *parallel trap*), without the need for the second subroutine.

We first state our result in the unconstrained setting, which is our main focus in this paper.

Theorem 4.1 *Let $d \geq 2$ and $f : \mathbb{R}^d \to [0, \infty)$ be such that ∇f is L-Lipschitz-continuous. For any $\varepsilon > 0$, the GFPT algorithm with starting point $x_0 \in \mathbb{R}^d$ returns an ε-stationary point using at most*

$$O(d)^{\frac{5d-1}{4}} \left(\frac{\sqrt{Lf(x_0)}}{\varepsilon} \right)^{d-1}$$

queries. In particular, for $d = 2$ the number of queries is $O(\sqrt{Lf(x_0)}/\varepsilon)$.

Note that if the co-domain is also bounded from above, i.e., $[0, 1]$ instead of $[0, \infty)$, then the bound becomes $(\sqrt{L}/\varepsilon)^{d-1}$ for any fixed d.

The GFPT algorithm also applies to the constrained setting, which is the setting in which the GFT algorithm was originally stated by Bubeck and Mikulincer [4].

Theorem 4.2 *Let $d \geq 2$ and $f : [0, 1]^d \to \mathbb{R}$ be such that ∇f is L-Lipschitz-continuous on $[0, 1]^d$. For any $\varepsilon > 0$, the GFPT algorithm returns an ε-KKT point (w.r.t. minimization) using at most*

$$O(d)^{\frac{5d-1}{4}} \left(\sqrt{\frac{L}{\varepsilon}}\right)^{d-1}$$

queries. In particular, for $d = 2$ the number of queries is $O(\sqrt{L/\varepsilon})$, and for $d = 3$ it is $O(L/\varepsilon)$.

In the next section we present a high-level overview of the algorithm. Then, we proceed with the formal presentation and proof for the unconstrained setting. Finally, we briefly mention how the algorithm can be adapted to the constrained setting.

4.1 Overview of the GFPT algorithm

For this overview we consider the 2-dimensional unconstrained setting. In other words, we assume that we have query access to a function $f : \mathbb{R}^2 \to [0, \infty)$, which has L-Lipschitz-continuous gradient ∇f. Recall that our goal is to find an ε-stationary point, i.e., a point $x \in \mathbb{R}^d$ such that $\|\nabla f(x)\|_2 \leq \varepsilon$.

Gradient flow. The notion of the *gradient flow* is very useful in order to understand the intuition behind the algorithm. Intuitively, the gradient flow is a continuous path that corresponds to the points that gradient descent would visit if it had an infinitesimally small step size. More formally, the gradient flow starting at x is the path $\gamma(t)$ that is the solution of the differential equation $\gamma'(t) = -\nabla f(\gamma(t))$ with initial condition $\gamma(0) = x$.

Initialization. Let $x_0 \in \mathbb{R}^d$ be some starting point. If $f(x_0) = 0$, then x_0 is a global minimum of the function and thus necessarily a stationary point, so we assume that $f(x_0) > 0$.

Consider the rectangle $R_0 = \{x \in \mathbb{R}^2 : \|x - x_0\|_\infty \leq 2f(x_0)/\varepsilon\}$. We claim that R_0 must contain an ε-stationary point of f. Indeed, assume that R_0 does not contain any ε-stationary points and consider the gradient flow starting at x_0. Then, as long as the gradient flow has not left R_0, the value of f along the gradient flow must decrease by a rate at least ε: if the gradient flow has traveled a distance ℓ, then the function value has decreased by at least $\varepsilon \cdot \ell$. In order to reach the boundary of R_0, the gradient flow must travel a distance at least $2f(x_0)/\varepsilon$ from x_0. But this means that, when the gradient flow reaches ∂R_0, the function value will be at most $f(x_0) - \varepsilon \cdot 2f(x_0)/\varepsilon = -f(x_0) < 0$, which is impossible! As a result, it follows that R_0 must in fact contain an ε-stationary point. See Fig. 28 for an illustration of this argument.

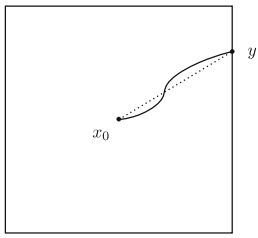


Fig. 28 An illustration of the rectangle R_0 and the gradient flow starting at x_0

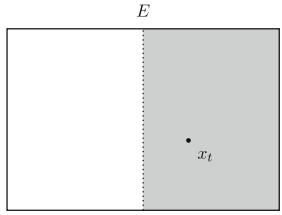


Fig. 29 A simple gradient flow trap. The gray region is the new smaller rectangle R_{t+1}

Invariant. The initialization step ensures that R_0 contains an ε-stationary point. However, R_0 is too large for us to locate such an ε-stationary point by brute force. Thus, the algorithm will seek to decrease the size of the rectangle at each iteration, while maintaining the invariant that it must contain a solution. After sufficiently many iterations, the rectangle will be small enough, so that the ε-stationary point can easily be found.

More formally, at each iteration t, the algorithm will maintain a rectangle R_t and a point $x_t \in R_t$, such that R_{t+1} is smaller than R_t by a constant fraction. The goal of the algorithm will be to maintain the following invariant: all points on the boundary of R_t are ε-unreachable from x_t. We say that a point y is ε-unreachable from a point x if

$$f(y) > f(x) - \varepsilon\|x - y\|_2.$$

The intuition for this definition is the following: if y is ε-unreachable from x, then the gradient flow starting at x cannot reach y, unless it encounters an ε-stationary point on the way. Thus, if (R_t, x_t) satisfies the invariant, then R_t must necessarily contain an ε-stationary point. As explained above, (R_0, x_0) satisfies the invariant, so the algorithm now has to find a way to decrease the size of R_t at each step while maintaining the invariant.

Simple gradient flow trap. Let us first consider a simple attempt at decreasing the size of R_t which unfortunately fails. Let E be the segment that cuts R_t in half (along its longest side). Furthermore, assume that we can somehow determine that all points $y \in E$ are ε-unreachable from x_t. In that case, we could simply set $x_{t+1} := x_t$ and let R_{t+1} be the half of R_t which contains x. It is easy to see that the invariant would be satisfied by (R_{t+1}, x_{t+1}). See Fig. 29 for an illustration.

There are multiple issues with this simple approach. One problem is that we have not said what happens when there exists some point $z \in E$ that is *not* ε-unreachable from x_t. A crucial observation, which we will also use later, is that in that case, all points $y \in \partial R_t$ are ε-unreachable from z. Indeed, we can combine the inequalities

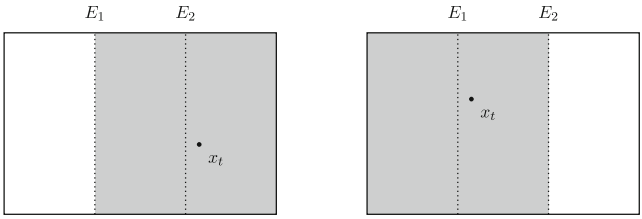


Fig. 30 The two possible cases when all points on S_1 and S_2 are ε-unreachable from x_t. The gray region represents the new smaller rectangle R_{t+1} in both cases

$f(z) \leq f(x_t) - \varepsilon\|x_t - z\|_2$ and $f(y) > f(x_t) - \varepsilon\|x_t - y\|_2$ to obtain

$$f(y) > f(x_t) - \varepsilon\|x_t - y\|_2 \geq f(z) + \varepsilon\|x_t - z\|_2 - \varepsilon\|x_t - y\|_2 \geq f(z) - \varepsilon\|z - y\|_2$$

i.e., y is ε-unreachable from z. This means that the update $x_{t+1} := z$ and $R_{t+1} := R_t$ would maintain the invariant. Unfortunately, this update would not decrease the size of the rectangle...

A more fundamental issue is: how can we determine whether all points $y \in E$ are ε-unreachable from x_t? This would require an infinite number of queries to f... Instead, we pick a sufficiently fine δ-discretization S of E and only query f on the points in S. What can we say now, if we find that all points in S are ε-unreachable from x_t? It turns out that as long as x_t is sufficiently far away from E, it follows that all points in E are ε'-unreachable from x_t, for some $\varepsilon' > \varepsilon$. As a result, our algorithm will have to allow for ε to increase (hopefully, only by a small amount!) from one iteration to the next one. In other words, we will also maintain a value ε_t at each iteration and the invariant will now be: $(R_t, x_t, \varepsilon_t)$ satisfies the invariant, if all points on the boundary of R_t are ε_t-unreachable from x_t. We will aim to ensure that ε_t does not increase too much, namely by at most a constant factor in *total* over all iterations. For this, we will have to pick δ to be sufficiently large, but also ensure that x_t is sufficiently far away from E. Unfortunately, this cannot be guaranteed using this simple gradient flow trap, which is why a parallel trap is needed, as already noted by Bubeck and Mikulincer [4]. In fact, the parallel will also allow us to take care of the first issue identified above.

The parallel trap. For concreteness, assume that the longest side of R_t is along the first coordinate, and that it has length r. Instead of considering a single segment E which cuts R_t in half, we consider segments E_1 and E_2 that cut R_t into three equal parts.

We pick sufficiently fine δ-discretizations S_1 and S_2 of E_1 and E_2 respectively. If all points on S_1 and S_2 are ε_t-unreachable from x_t, then it is easy to see that we can remove a third of R_t while maintaining the invariant without changing x_t. If x_t lies in the right half of R_t, then we remove everything on the left of E_1. Otherwise, we remove everything on the right of E_2. In both cases we let $x_{t+1} := x_t$. See Fig. 30 for an illustration. Note that in both cases x_t cannot be arbitrarily close to the new boundary of R_{t+1}: the distance is always at least $r/6$. This is important to ensure that we can control how much larger ε_{t+1} is than ε_t.

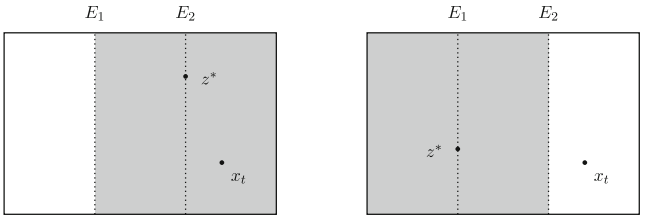


Fig. 31 The two possible cases when there exist points on S_1 and S_2 that are not ε-unreachable from x_t. The gray region represents the new smaller rectangle R_{t+1} in both cases

If, on the other hand, there exist points z on S_1 or S_2 that are not ε_t-unreachable from x, then let z^* denote such a point with minimal value $f(z^*)$. For concreteness, assume that z^* lies on S_1. By the crucial observation presented in the simple gradient flow trap attempt above, we know that all points on ∂R_t are ε_t-unreachable from z^*. Thus, we will set $x_{t+1} := z^*$, but we will also let R_{t+1} be R_t with everything on the right of E_2 removed. See Fig. 31 for an illustration. This will ensure that the size of the rectangle decreases, but we still need to argue that this maintains the invariant. We will show that all points on S_2 are ε_t-unreachable from z^*. Then, since z^* is sufficiently far away from E_2, we will again be able to argue that ε_{t+1} is not much larger than ε_t. First, consider points in S_2 that are ε_t-unreachable from x_t. By the crucial observation, these points are also ε_t-unreachable from z^*. It remains to consider points $z \in S_2$ that are not ε_t-unreachable from x_t. But by construction of z^* we have $f(z^*) \leq f(z)$ for all these points z. Together with the fact that $\|z^* - z\|_2 > 0$ (since $z^* \in E_1$ and $z \in E_2$) this implies

$$f(z) \geq f(z^*) > f(z^*) - \varepsilon_t \|z^* - z\|_2$$

i.e., z is ε_t-unreachable from z^*.

Parameters. In the formal analysis of the algorithm we show that it is possible to pick the size of the δ-nets in each step so as to balance out the following two objectives: (i) δ is small enough so that, in total over all iterations, ε only increases by a constant factor, and (ii) δ is large enough, so that the total number of queries remains low, namely $O(\sqrt{Lf(x_0)}/\varepsilon)$ in the two-dimensional case.

4.2 Formal presentation of GFPT

Let $d \geq 2$ and let $f : \mathbb{R}^d \to [0, \infty)$ be such that ∇f is L-Lipschitz-continuous. We begin with some definitions and technical lemmas.

Definition 4.1 For $x, y \in \mathbb{R}^d$ and $\varepsilon > 0$ we say that y *is* ε*-unreachable from* x if the following holds:

$$f(y) > f(x) - \varepsilon \|x - y\|_2.$$

Intuitively, y is ε-unreachable from x if the value $f(y)$ is too high with respect to $f(x)$ so that y cannot be reached by following the "gradient flow" starting at x without encountering an ε-stationary point along the way. We now formalize this intuition.

Definition 4.2 A k-dimensional hyperrectangle R in $\mathbb{R}^d$ is a set of the form $R = [a_1, b_1] \times \cdots \times [a_d, b_d]$, where $a_i \leq b_i$ for all $i \in [d]$, and $|\{i \in [d] : a_i < b_i\}| = k$. When $k = d$, we also say that R is full-dimensional.

The following lemma is essentially a more refined version of a corresponding result in [4] and uses the same proof technique.

Lemma 7 *Let R be a full-dimensional hyperrectangle in $\mathbb{R}^d$ and x a point in R. For any $\varepsilon > 0$, if all $y \in \partial R$ are ε-unreachable from x, then R contains an ε-stationary point of f.*

Proof The gradient flow for f starting at x is defined as the solution $\gamma(t)$ to the differential equation

$$\gamma'(t) = -\nabla f(\gamma(t))$$

with the initial condition $\gamma(0) = x$. The gradient flow is a curve that starts at x and follows the direction of steepest descent as given by the negative of the gradient ∇f. Since ∇f is Lipschitz-continuous, the existence and uniqueness of a solution $\gamma : [0, \infty) \to \mathbb{R}^d$ is guaranteed by standard tools from the theory of differential equations.

Define $\psi : t \mapsto f(\gamma(t))$ and note that ψ is continuously differentiable with derivative $\psi'(t) = \langle \nabla f(\gamma(t)), \gamma'(t) \rangle$. For any $T \geq 0$ we can write

$$\begin{aligned} f(\gamma(T)) - f(\gamma(0)) &= \psi(T) - \psi(0) = \int_0^T \psi'(t)dt \\ &= \int_0^T \langle \nabla f(\gamma(t)), \gamma'(t) \rangle dt \\ &= \int_0^T \langle \nabla f(\gamma(t)), -\nabla f(\gamma(t)) \rangle dt \\ &= -\int_0^T \|\nabla f(\gamma(t))\|_2^2 dt. \end{aligned}$$

Assume towards a contradiction that the hyperrectangle R does not contain any ε-stationary points of f. In other words, $\|\nabla f(z)\|_2 > \varepsilon$ for all $z \in R$. Now consider any $T \geq 0$ such that $\gamma(t) \in R$ for all $t \in [0, T]$. Then we have that

$$f(\gamma(T)) - f(\gamma(0)) = -\int_0^T \|\nabla f(\gamma(t))\|_2^2 dt \leq -T\varepsilon^2.$$

Since f is continuous and R is compact, f is bounded on R and thus there exists a maximal T such that $\gamma(t) \in R$ for all $t \in [0, T]$. In particular, $y := \gamma(T)$ lies on the boundary ∂R. Furthermore, since the length of the curve can be computed as $\int_0^T \|\gamma'(t)\|_2 dt$, we deduce that $\int_0^T \|\gamma'(t)\|_2 dt \geq \|\gamma(0) - \gamma(T)\|_2 = \|x - y\|_2$. Finally,

we obtain

$$\begin{aligned} f(y) - f(x) &= f(\gamma(T)) - f(\gamma(0)) = -\int_0^T \|\nabla f(\gamma(t))\|_2^2 dt \\ &\leq -\varepsilon \int_0^T \|\nabla f(\gamma(t))\|_2 dt \\ &= -\varepsilon \int_0^T \|\gamma'(t)\|_2 dt \\ &\leq -\varepsilon \|x - y\|_2. \end{aligned}$$

But this means that $y \in \partial R$ is not ε-unreachable from x, a contradiction. As a result, R must necessarily contain some ε-stationary point. □

Corollary 2 *Let $R = [a_1, b_1] \times \cdots \times [a_d, b_d]$ be a full-dimensional hyperrectangle in $\mathbb{R}^d$ and x a point in R. For any $\varepsilon > 0$, if all $y \in \partial R$ are $(\varepsilon/2)$-unreachable from x, and $\max_i(b_i - a_i) \leq \frac{\varepsilon}{2\sqrt{d}L}$, then x is an ε-stationary point of f.*

Proof By Lemma 7, R contains an $\varepsilon/2$-stationary point of f, i.e., there exists $y \in R$ with $\|\nabla f(y)\|_2 \leq \varepsilon/2$. Since the gradient of f is L-Lipschitz-continuous, we have that

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2 \leq L\sqrt{d}\|x - y\|_\infty \leq L\sqrt{d} \max_i(b_i - a_i) \leq \varepsilon/2$$

and thus $\|\nabla f(x)\|_2 \leq \varepsilon$. □

The algorithm will make use of δ-nets with some nice properties.

Definition 4.3 (*nice δ-net*) Let $\delta > 0$ and let R be a k-dimensional hyperrectangle in $\mathbb{R}^d$. A set of points $S \subseteq R$ is a *nice δ-net* of R if for any face F of R it holds that $S \cap F$ is a δ-net of F, i.e., for all $y \in F$ there exists $z \in S \cap F$ with $\|z - y\|_2 \leq \delta$.

It is not hard to construct nice δ-nets of reasonable size. We include the following construction for completeness.

Lemma 8 *Let $R = [a_1, b_1] \times \cdots \times [a_d, b_d]$ be a k-dimensional hyperrectangle in $\mathbb{R}^d$. Then, for any $\delta > 0$, there exists a nice δ-net S of R with $|S| = \prod_{i=1}^d (\lceil \sqrt{k}(b_i - a_i)/2\delta \rceil + 1)$. In particular, if R is $(d-1)$-dimensional, then we have $|S| \leq (\sqrt{d}r/2\delta)^{d-1}$ for any r satisfying $r \geq \max_i(b_i - a_i)$ and $r \geq 8\sqrt{d}\delta$.*

Proof For each $i \in [d]$, we construct a set $S_i \subseteq [a_i, b_i]$ as follows. If $a_i = b_i$, we let $S_i := \{a_i\}$. Otherwise, namely when $a_i < b_i$, we partition the interval $[a_i, b_i]$ into $\lceil \sqrt{k}(b_i - a_i)/2\delta \rceil$ intervals of equal length, and we let $S_i \subset [a_i, b_i]$ denote the set of endpoints of these intervals. Note that we always have $|S_i| = \lceil \sqrt{k}(b_i - a_i)/2\delta \rceil + 1$ and $\{a_i, b_i\} \subseteq S_i$. Furthermore, by construction, the distance between subsequent points in S_i is at most $2\delta/\sqrt{k}$. Thus, for any $p \in [a_i, b_i]$ there exists $q \in S_i$ such that $|p - q| \leq \delta/\sqrt{k}$.

We now prove that $S = S_1 \times \cdots \times S_d$ is a nice δ-net of R. Consider any point $y \in R$. We construct $z \in S$ as follows: for each $i \in [d]$, let $z_i = \operatorname{argmin}_{p \in S_i} |p - y_i|$ (pick an arbitrary minimizer, if it is not unique). First of all, note that we indeed have $z \in S$ by construction of S. Furthermore, by construction of S_i, we have that $|z_i - y_i| \leq \delta/\sqrt{k}$ for all $i \in [d]$. This implies that $\|z - y\|_2 \leq \delta$, since R is k-dimensional and thus z and y disagree on at most k coordinates. Finally, let F be any face of R that contains y. Since $\{a_i, b_i\} \subseteq S_i$, the construction ensures that $z_i = y_i$ whenever $y_i \in \{a_i, b_i\}$. Thus, we also have $z \in F$. In other words, $S \cap F$ is a δ-net of F.

Now consider the case where R is $(d-1)$-dimensional and where r is some value satisfying $r \geq \max_i(b_i - a_i)$ and $r \geq 8\sqrt{d}\delta$. Using the fact that $b_i - a_i \leq r$ for all i, and that $b_j - a_j = 0$ for some j, we obtain

$$\begin{aligned} |S| = \prod_{i=1}^{d} \left(\left\lceil \frac{\sqrt{d-1}(b_i - a_i)}{2\delta} \right\rceil + 1 \right) \\ \leq \left(\left\lceil \frac{\sqrt{d-1}r}{2\delta} \right\rceil + 1 \right)^{d-1} \leq \left(\frac{\sqrt{d-1}r}{2\delta} + 2 \right)^{d-1} \\ \leq \left(\frac{\sqrt{d}r}{2\delta} \right)^{d-1} \end{aligned}$$

where the last inequality follows from

$$\begin{aligned} \frac{\sqrt{d}r}{2\delta} - \frac{\sqrt{d-1}r}{2\delta} = \frac{(\sqrt{d} - \sqrt{d-1})r}{2\delta} &\geq 4\sqrt{d}(\sqrt{d} - \sqrt{d-1}) \\ &\geq 2(\sqrt{d} + \sqrt{d-1})(\sqrt{d} - \sqrt{d-1}) \\ &\geq 2(d - (d-1)) = 2 \end{aligned}$$

where we used $r \geq 8\sqrt{d}\delta$. □

Before giving the algorithm we prove the following technical lemma, which will be important for the proof of correctness.

Lemma 9 *Let E be a $(d-1)$-dimensional hyperrectangle of $\mathbb{R}^d$ and let S be a nice δ-net of E. Let $\varepsilon > 0$ and let $x \in \mathbb{R}^d$ with $\ell := \operatorname{dist}(x, E) > 0$. Then, if all $z \in S$ are ε-unreachable from x, it follows that all $y \in E$ are ε'-unreachable from x, where*

$$\varepsilon' = \varepsilon + \frac{\delta^2}{2\ell}(L + 2\varepsilon/\ell).$$

Here $\operatorname{dist}(x, E) := \min_{y \in E} \|x - y\|_2$.

Proof Consider the function $\phi : E \to \mathbb{R}$ defined by

$$\phi(y) = f(y) - f(x) + \varepsilon\|x - y\|_2.$$

For all $z \in S$ we have that $\phi(z) > 0$, since z is ε-unreachable from x. We will show that for all $y \in E$ we have $\phi(y) > -\frac{\delta^2}{2}(L + 2\varepsilon/\ell)$. Note that this suffices to prove that all $y \in E$ are ε'-unreachable from x, since

$$\begin{aligned} f(y) = f(x) - \varepsilon\|x - y\|_2 + \phi(y) &> f(x) - \varepsilon\|x - y\|_2 - \frac{\delta^2}{2}(L + 2\varepsilon/\ell) \\ &\geq f(x) - \left(\varepsilon + \frac{\delta^2}{2\ell}(L + 2\varepsilon/\ell)\right)\|x - y\|_2 \\ &= f(x) - \varepsilon'\|x - y\|_2 \end{aligned}$$

where we used $\|x - y\|_2 \geq \text{dist}(x, E) = \ell$.

Fix some $y^* \in \text{argmin}_{y \in E}\, \phi(y)$. In the remainder of this proof, we show that $\phi(y^*) > -\frac{\delta^2}{2}(L + 2\varepsilon/\ell)$. Let F denote the smallest face of E that contains y^*. If F is 0-dimensional, i.e., a corner of the hyperrectangle E, then by the definition of a nice δ-net it follows that $S \cap F \neq \emptyset$ and thus $y^* \in S$. In particular, $\phi(y^*) > 0 \geq -\frac{\delta^2}{2}(L + 2\varepsilon/\ell)$. Now consider the case where F is k-dimensional for some $k \in \{1, 2, \ldots, d - 1\}$. Note that y^* cannot lie on the boundary of F; otherwise, F would not be the smallest face of E containing y^*. Furthermore, ϕ is continuously differentiable on E, and thus also on the face F, since f is continuously differentiable on E and since $\|x - y\|_2 \geq \text{dist}(x, E) > 0$ for all $y \in E$. Hence, given that y^* is a minimum of ϕ on F, and that it lies in the interior of F, we must have that $[\nabla\phi(y^*)]_F = 0$. Here $[\cdot]_F \in \mathbb{R}^k$ denotes the restriction to the k coordinates that are not fixed on the face F.

Since S is a nice δ-net of E, it follows that $S \cap F$ is a δ-net of F. As a result, there exists $z \in S \cap F$ with $\|z - y^*\|_2 \leq \delta$. Let ψ denote the function ϕ on the segment $[y^*, z]$ in F, parameterized as $\psi : [0, 1] \to \mathbb{R}, t \mapsto \phi(y^* + t(z - y^*))$. Note that ψ is continuously differentiable and $\psi'(t) = \langle z - y^*, \nabla\phi(y^* + t(z - y^*))\rangle$. Using the fact that (i) $\nabla\phi(y) = \nabla f(y) + \varepsilon(y - x)/\|y - x\|_2$, (ii) ∇f is L-Lipschitz-continuous, and (iii) $\|x - y\|_2 \geq \text{dist}(x, E) = \ell$ for all $y \in E \supseteq F$, it can be shown that $\psi'(t)$ is L'-Lipschitz-continuous for $L' = \|z - y^*\|_2^2(L + 2\varepsilon/\ell) \leq \delta^2(L + 2\varepsilon/\ell)$.

Furthermore, $\psi'(0) = \langle z - y^*, \nabla\phi(y^*)\rangle = \langle [z - y^*]_F, \nabla^F\phi(y^*)\rangle = 0$, where we used the fact that z and y^* both lie in F (and thus have the same value in all coordinates that are not fixed in F) and $\nabla^F\phi(y^*) = 0$. As a result, for any $t \in [0, 1]$, we have

$$|\psi'(t)| = |\psi'(t) - \psi'(0)| \leq L'|t - 0| \leq \delta^2(L + 2\varepsilon/\ell)t.$$

Now, we can write

$$\phi(z) - \phi(y^*) = \psi(1) - \psi(0) = \int_0^1 \psi'(t)dt \leq \int_0^1 \delta^2(L + 2\varepsilon/\ell)tdt = \frac{\delta^2}{2}(L + 2\varepsilon/\ell)$$

which yields $\phi(y^*) \geq \phi(z) - \frac{\delta^2}{2}(L + 2\varepsilon/\ell) > -\frac{\delta^2}{2}(L + 2\varepsilon/\ell)$ as desired. □

We are now ready to present the full algorithm.

Algorithm 1: The Gradient Flow Parallel Trap (GFPT) Algorithm

input : $\varepsilon > 0$, $L > 0$, $d \geq 2$, $x_0 \in \mathbb{R}^d$, query access to $f : \mathbb{R}^d \to [0, +\infty)$ with L-Lipschitz ∇f

output : an ε-stationary point of f

Set $t := 0$ and $\varepsilon_0 := \varepsilon/4$. Let $C_1 := 75\sqrt{d}$ and $C_2 := 16d$.

Initialize hyperrectangle $R_0 = [a_1, b_1] \times \cdots \times [a_d, b_d]$ by setting

$$R_0 := \{x \in \mathbb{R}^d : \|x - x_0\|_\infty \leq 2f(x_0)/\varepsilon_0\}.$$

while $r_t := \max_i(b_i - a_i) > \frac{\varepsilon}{2\sqrt{d}L}$ **do**

  Pick $j \in \operatorname{argmax}_i(b_i - a_i)$ (arbitrarily).

  Set $\delta_t := \sqrt{\frac{\varepsilon}{C_1 C_2 L} r_t (3/4)^{\lfloor t/d \rfloor}}$ and query f on a nice δ_t-net S_1 of E_1 and S_2 of E_2, where

$$E_1 = [a_1, b_1] \times \cdots \times \{a_j + r_t/3\} \times \cdots \times [a_d, b_d],$$

$$E_2 = [a_1, b_1] \times \cdots \times \{b_j - r_t/3\} \times \cdots \times [a_d, b_d].$$

  Let $S^* := \{z \in S_1 \cup S_2 : f(z) \leq f(x_t) - \varepsilon_t \|x_t - z\|_2\}$ be the set of all points on the nets that are *not* ε_t-unreachable from x_t.

  if $S^* = \emptyset$ **then** $x_{t+1} := x_t$ **else** $x_{t+1} := z^* \in \operatorname{argmin}_{z \in S^*} f(z)$.

  if $[x_{t+1}]_j \geq a_j + r_t/2$ **then** $a_j := a_j + r_t/3$ **else** $b_j := b_j - r_t/3$.

  Set $\varepsilon_{t+1} := \varepsilon_t + \frac{\varepsilon (3/4)^{\lfloor t/d \rfloor}}{C_2}$ and update $t := t + 1$.

Output x_t.

Note that at each iteration of the algorithm, the length of the rectangle R along some dimension j decreases by a factor $2/3$. Since we always pick a dimension j along which the rectangle R has maximal length, and since we stop as soon as the length in each dimension is at most $\varepsilon/2\sqrt{d}L$, the algorithm terminates after $T = d\lceil \log_{3/2}(r_0/(\varepsilon/2\sqrt{d}L)) \rceil = d\lceil \log_{3/2}(16\sqrt{d}f(x_0)L/\varepsilon^2) \rceil$ iterations.

We first argue about the correctness of the algorithm.

Lemma 10 *The output of the algorithm x_T is an ε-stationary point of f.*

Proof First of all, note that if $f(x_0) = 0$, then the algorithm outputs x_0 and x_0 is a global minimum of f and thus a stationary point. Thus, for the rest of this proof we assume that $f(x_0) > 0$.

Let R_t denote the hyperrectangle at the beginning of iteration t. In particular, $R_0 = R$. We will show that the algorithm satisfies the following invariant at every iteration: all points $y \in \partial R_t$ are ε_t-unreachable from x_t.

Let us first see why this invariant implies the correctness of the algorithm. At the last iteration T of the algorithm, we can bound

$$\begin{aligned}\varepsilon_T = \varepsilon_{T-1} + \frac{\varepsilon(3/4)^{\lfloor (T-1)/d \rfloor}}{C_2} = \cdots = \varepsilon_0 + \sum_{t=0}^{T-1} \frac{\varepsilon(3/4)^{\lfloor t/d \rfloor}}{C_2} &\leq \varepsilon_0 + \sum_{t=0}^{\infty} \frac{\varepsilon(3/4)^{\lfloor t/d \rfloor}}{C_2} \\ &= \varepsilon_0 + \frac{\varepsilon}{16d} \cdot d \cdot \sum_{i=0}^{\infty} (3/4)^i \\ &\leq \varepsilon/4 + \varepsilon/4 = \varepsilon/2.\end{aligned}$$

Thus, by using the invariant for $t = T$, all $y \in \partial R_T$ are $(\varepsilon/2)$-unreachable from x_T. Since the rectangle R_T has side-length at most $\varepsilon/2\sqrt{d}L$, by Corollary 2 x_T is an ε-stationary point of f.

It remains to prove the invariant. Note that by construction of $R = R_0$ the invariant holds at $t = 0$. Indeed, for all $y \in \partial R_0$, we have $\|x_0 - y\|_2 \geq \|x_0 - y\|_\infty = 2f(x_0)/\varepsilon_0$ and together with the fact that $f(y) \geq 0$ this yields

$$f(y) \geq 0 \geq 2f(x_0) - \varepsilon_0 \|x_0 - y\|_2 > f(x_0) - \varepsilon_0 \|x_0 - y\|_2$$

where we used the fact that $f(x_0) > 0$.

Now consider any t such that the invariant holds for iteration t. We will show that it also holds for iteration $t + 1$.

There are two cases to consider, depending on whether $S^* = \emptyset$ or not. First consider the case where $S^* = \emptyset$. In that case, the algorithm sets $x_{t+1} := x_t$. Note that all points $y \in \partial R_{t+1} \cap \partial R_t$ are ε_t-unreachable from x_t by the invariant, and so also ε_{t+1}-unreachable from x_{t+1}, since $\varepsilon_{t+1} \geq \varepsilon_t$. It remains to show that all points in $\partial R_{t+1} \backslash \partial R_t$ are also ε_{t+1}-unreachable from x_t. By construction of the algorithm it holds that $\partial R_{t+1} \backslash \partial R_t \subseteq E_i$, where $i \in \{1, 2\}$ is such that $\ell := \text{dist}(x_t, E_i) \geq r_t/6$. Furthermore, since $S^* = \emptyset$, it follows that all points on the nice δ_t-net S_i of E_i are ε_t-unreachable from x_t. As a result, we can apply Lemma 9 to deduce that all $y \in E_i$ are ε'-unreachable from x, where

$$\begin{aligned}\varepsilon' &= \varepsilon_t + \frac{\delta_t^2}{2\ell}(L + 2\varepsilon_t/\ell) \leq \varepsilon_t + \frac{\delta_t^2}{2\ell}(L + 24\sqrt{d}L) \leq \varepsilon_t + \frac{3\delta_t^2}{r_t}(L + 24\sqrt{d}L) \\ &\leq \varepsilon_t + \frac{75\sqrt{d}L\delta_t^2}{r_t} \\ &= \varepsilon_t + \frac{C_1 L \delta_t^2}{r_t} \\ &= \varepsilon_t + \frac{\varepsilon(3/4)^{\lfloor t/d \rfloor}}{C_2} \\ &= \varepsilon_{t+1}\end{aligned}$$

where we used $\ell \geq r_t/6 \geq \varepsilon/12\sqrt{d}L$, $\varepsilon_t \leq \varepsilon$ and $\delta_t := \sqrt{\frac{\varepsilon}{C_1C_2L} r_t (3/4)^{\lfloor t/d \rfloor}}$. Thus the invariant holds in the first case.

Now consider the case where $S^* \neq \emptyset$. In that case, the algorithm sets $x_{t+1} := z^*$, where $z^* \in \operatorname{argmin}_{z \in S^*} f(z)$. As before, by the invariant we know that all points in $\partial R_{t+1} \cap \partial R_t$ are ε_t-unreachable from x_t. Furthermore, since $z^* \in S^*$, we have that $f(z^*) \leq f(x_t) - \varepsilon_t \|x_t - z^*\|_2$. As a result, using the triangle inequality, we obtain that for any $y \in \partial R_{t+1} \cap \partial R_t$

$$f(y) > f(x_t) - \varepsilon_t \|x_t - y\|_2 \geq f(z^*) + \varepsilon_t \|x_t - z^*\|_2 - \varepsilon_t \|x_t - y\|_2 \geq f(z^*) - \varepsilon_t \|z^* - y\|_2$$

i.e., y is ε_t-unreachable from z^*. Since $\varepsilon_{t+1} \geq \varepsilon_t$, it follows that all points in $\partial R_{t+1} \cap \partial R_t$ are ε_{t+1}-unreachable from z^*. It remains to show that all points in $\partial R_{t+1} \setminus \partial R_t$ are also ε_{t+1}-unreachable from $z^* = x_{t+1}$. By construction of the algorithm it holds that $\partial R_{t+1} \setminus \partial R_t \subseteq E_i$, where $i \in \{1, 2\}$ is such that $\operatorname{dist}(z^*, E_i) = r_t/3$. We first show that all points in S_i are ε_t-unreachable from z^*. For all points $z \in S_i \setminus S^*$, note that they are ε_t-unreachable from x_t, and thus also ε_t-unreachable from z^* by using the triangle inequality as we did earlier for the points $y \in \partial R_{t+1} \cap \partial R_t$. For all points $z \in S_i \cap S^*$, we must have $f(z) \geq f(z^*)$ since we picked $z^* \in \operatorname{argmin}_{z \in S^*} f(z)$. But since $z \in S_i \subset E_i$ and $\operatorname{dist}(z^*, E_i) = r_t/3$, we must have $\|z^* - z\|_2 \geq r_t/3 > 0$, and thus $f(z) > f(z^*) - \varepsilon_t \|z^* - z\|_2$, i.e., z is ε_t-unreachable from z^*. Thus, all points in S_i are ε_t-unreachable from z^*. Finally, just as in the previous case, we can use Lemma 9 to deduce from this that all points in E_i are ε_{t+1}-unreachable from z^*. In particular, note that we have $\operatorname{dist}(z^*, E_i) = r_t/3 \geq r_t/6$ so the same arguments can indeed be applied. As a result, the invariant holds in the second case as well. □

Lemma 11 *For any $d \geq 2$, the number of queries to f performed by the algorithm is*

$$O(d)^{\frac{5d-1}{4}} \left(\frac{\sqrt{Lf(x_0)}}{\varepsilon} \right)^{d-1}$$

In particular, for $d = 2$ the number of queries is $O(\sqrt{Lf(x_0)}/\varepsilon)$.

Proof Let q_t denote the number of queries performed by the algorithm in iteration t. Then, the total number of queries performed by the algorithm is $Q = \sum_{t=0}^{T-1} q_t$. In iteration t the algorithm queries the value of f on two nice δ_t-nets S_1 and S_2 of E_1 and E_2 respectively. Since E_1 and E_2 are $(d-1)$-dimensional hyperrectangles with side-length bounded by r_t, using Lemma 8 we obtain that

$$q_t \leq 2 \left(\frac{\sqrt{d} r_t}{2\delta_t} \right)^{d-1}.$$

Note that Lemma 8 also requires $r_t \geq 8\sqrt{d}\delta_t$. This indeed holds for all $t \in \{0, 1, \ldots, T-1\}$, since

$$\frac{r_t}{\delta_t} = \sqrt{\frac{C_1C_2Lr_t}{\varepsilon(3/4)^{\lfloor t/d \rfloor}}} \geq \sqrt{\frac{C_1C_2Lr_t}{\varepsilon}} \geq \sqrt{\frac{C_1C_2}{2\sqrt{d}}} \geq 8\sqrt{d}$$

where we used $r_t \geq \varepsilon/2\sqrt{d}L$. As a result, the total number of queries Q can be bounded as follows

$$Q = \sum_{t=0}^{T-1} q_t \leq \sum_{t=0}^{T-1} 2\left(\frac{\sqrt{d}r_t}{2\delta_t}\right)^{d-1} = 2\sum_{t=0}^{T-1}\left(\sqrt{\frac{dC_1C_2Lr_t}{4\varepsilon(3/4)^{\lfloor t/d\rfloor}}}\right)^{d-1}$$
$$\leq 2\left(\frac{dC_1C_2}{4}\right)^{\frac{d-1}{2}}\left(\frac{Lr_0}{\varepsilon}\right)^{\frac{d-1}{2}}\sum_{t=0}^{T-1}\left(\frac{(2/3)^{\lfloor t/d\rfloor}}{(3/4)^{\lfloor t/d\rfloor}}\right)^{\frac{d-1}{2}}$$

where we used $r_t = r_0(2/3)^{\lfloor t/d\rfloor}$. Furthermore, we can bound

$$\sum_{t=0}^{T-1}\left(\frac{(2/3)^{\lfloor t/d\rfloor}}{(3/4)^{\lfloor t/d\rfloor}}\right)^{\frac{d-1}{2}} \leq d\sum_{i=0}^{\infty}\left(\left(\frac{8}{9}\right)^i\right)^{\frac{d-1}{2}} = \frac{d}{1-(8/9)^{(d-1)/2}} \leq 18d$$

since $d \geq 2$. Thus, we obtain

$$Q \leq O(d)^{\frac{5d-1}{4}}\left(\frac{Lr_0}{\varepsilon}\right)^{\frac{d-1}{2}}.$$

Finally, using the fact that $r_0 = 2f(x_0)/\varepsilon_0 = 8f(x_0)/\varepsilon$ we have

$$Q \leq O(d)^{\frac{5d-1}{4}}\left(\frac{\sqrt{Lf(x_0)}}{\varepsilon}\right)^{d-1}.$$

□

Remark 3 The analysis shows that we could have replaced the constant 3/4 by any other constant $\alpha \in (2/3, 1)$. In that case one would also have to modify the values of C_1 and C_2 accordingly.

4.3 Adapting GFPT to the constrained setting

In this section we briefly discuss how the algorithm can be adapted to the problem of finding stationary points in a compact domain.

To be more specific, let us consider the minimization problem for a function $f : [0, 1]^d \to \mathbb{R}$ with an L-Lipschitz-continuous gradient on $[0, 1]^d$. Note that unlike in the unconstrained case, we no longer assume (i) that the function is lower bounded, and (ii) that we have a starting point x_0. The goal is to find an ε-stationary point for the constrained setting, also known as an ε-KKT point, i.e., a point $x \in [0, 1]^d$ such that $\|g(x)\|_2 \leq \varepsilon$, where g is the *projected gradient* of f on $[0, 1]^d$. For the minimization setting, the projected gradient $g : [0, 1]^d \to \mathbb{R}^d$ of f on $[0, 1]^d$ is defined as

$$g_i(x) = \begin{cases} \min\{0, [\nabla f(x)]_i\} & \text{if } x_i = 0 \\ [\nabla f(x)]_i & \text{if } x_i \in (0, 1) \\ \max\{0, [\nabla f(x)]_i\} & \text{if } x_i = 1 \end{cases}$$

The modifications to the algorithm are very mild, namely:

- **Initialization:** We initialize x_0 to be an arbitrary point in $[0,1]^d$, e.g., $x = (1/2, \dots, 1/2)$. Furthermore, instead of initializing

$$R_0 := \{x \in \mathbb{R}^d : \|x - x_0\|_\infty \leq 2f(x_0)/\varepsilon_0\}$$

  we initialize

$$R_0 := [0,1]^d.$$

- **Extraction of a solution:** Instead of outputting x_T, we check the 2^d corners of R_T until we find one with $\|g(y)\|_2 \leq \varepsilon$. We can use $O(d)$ queries to f to compute a sufficiently good approximation of ∇f and thus g at any point.

The rest of the algorithm is unchanged. The analysis of the algorithm is mostly the same. Here we highlight the main differences.

- **Correctness:** The invariant is modified to say that at every iteration t: for all points y that lie on a facet E of R_t with $E \nsubseteq \partial[0,1]^d$, we have that y is ε_t-unreachable from x_t. In other words, the old invariant still holds, but only on facets of R_t that are not part of the boundary of the domain $[0,1]^d$. The proof of this new invariant is essentially identical to the proof for the old invariant.
  This new invariant, together with a modified version of the proof of Lemma 7 (where we consider the piecewise differentiable *projected* gradient flow defined by $\gamma'(t) = -g(\gamma(t))$) yields: there exists a point $x \in R_T$ with $\|g(x)\|_2 \leq \varepsilon/2$. A simple argument then shows that any corner y of the smallest face of R_T containing x must satisfy $\|g(y)\|_2 \leq \varepsilon$. Thus, the algorithm indeed returns an ε-stationary point.
- **Running time:** The analysis of the number of queries used by the algorithm is identical to the unconstrained version, up to the point where the bound

$$O(d)^{\frac{5d-1}{4}} \left(\frac{Lr_0}{\varepsilon}\right)^{\frac{d-1}{2}}$$

  is derived. Then, using the fact that we now have $r_0 = 1$ (instead of $r_0 = 2f(x_0)/\varepsilon_0$), we obtain the bound

$$O(d)^{\frac{5d-1}{4}} \left(\sqrt{\frac{L}{\varepsilon}}\right)^{d-1}.$$

A Definition of search problems and reductions

Definition A.1 (*Search Problems* - 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 $\perp$ 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. The class TFNP is defined as the set of all FNP problems $\mathcal{Q}$ that are *total*, i.e., for all $\boldsymbol{x} \in \{0,1\}^*$ there exists $\boldsymbol{y} \in \{0,1\}^*$ with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{Q}$.

Definition A.2 (*Polynomial-Time Reductions*) A search problem P_1 is *polynomial-time reducible* to P_2 if there exist polynomial-time computable functions $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ 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 $g(\boldsymbol{x}, \boldsymbol{y})$ is a solution to P_1 on input $\boldsymbol{x}$.

We also refer to [2] for a formal definition of black-box reductions.

A.1 Complete definition of the problem STATIONARY

Definition A.3 STATIONARY:
Input:

- precision parameter $\varepsilon > 0$,
- Turing machines $\mathcal{C}_f$ and $\mathcal{C}_{\nabla f}$ representing $f : \mathbb{R}^d \to \mathbb{R}$ and $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$,
- a boundedness constant $B > 0$, and a smoothness constant $L > 0$.

Goal: Find $x^\star \in \mathbb{R}^d$ such that $\|\nabla f(x^\star)\|_2 \leq \varepsilon$.

Alternatively, we also accept one of the following violations as a solution:

- (f or ∇f is not L-Lipschitz) $x, y \in \mathbb{R}^d$ such that

$$|f(x) - f(y)| > L\|x - y\|_2 \quad \text{or} \quad \|\nabla f(x) - \nabla f(y)\|_2 > L\|x - y\|_2,$$

- (f is not B-bounded) $x \in \mathbb{R}^d$ such that

$$|f(x)| > B,$$

- (∇f is not the gradient of f) $x, y \in \mathbb{R}^d$ that contradict Taylor's theorem, i.e.,

$$\left|f(y) - f(x) - \langle \nabla f(x), y - x\rangle\right| > \frac{L}{2}\|y - x\|_2^2.$$

Acknowledgements We thank the COLT and Mathematical Programming reviewers for helpful suggestions, and Takashi Ishizuka for providing feedback on an earlier version.

Funding AH was supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00026. MZ was supported by the Army Research Office (ARO) under contract

W911NF-17-1-0304 as part of the collaboration between US DOD, UK MOD and UK Engineering and Physical Research Council (EPSRC) under the Multidisciplinary University Research Initiative (MURI).

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Babichenko, Y., Rubinstein, A.: Settling the complexity of Nash equilibrium in congestion games. Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC), 1426–1437, (2021). https://doi.org/10.1145/3406325.3451039
2. Beame, P., Cook, S., Edmonds, J., Impagliazzo, R., Pitassi, T.: The relative complexity of NP search problems. J. Comput. Syst. Sci. **57**(1), 3–19 (1998). https://doi.org/10.1006/jcss.1998.1575
3. Boyd, S.: Stephen P Boyd, and Lieven Vandenberghe. Cambridge University Press, Convex optimization (2004)
4. Bubeck, S., Mikulincer, D.: How to trap a gradient flow. In Proceedings of the 33rd Conference on Learning Theory (COLT), 940–960, (2020). URL http://proceedings.mlr.press/v125/bubeck20b.html
5. Buresh-Oppenheim, J., Morioka, T.: Relativized NP search problems and propositional proof systems. In Proceedings of the 19th IEEE Conference on Computational Complexity (CCC), 54–67, (2004). https://doi.org/10.1109/CCC.2004.1313795
6. Buss, S.R., Johnson, A.S.: Propositional proofs and reductions between NP search problems. Ann. Pure Appl. Logic **163**(9), 1163–1182 (2012). https://doi.org/10.1016/j.apal.2012.01.015
7. Carmon, Y., Duchi, J.C., Hinder, O., Sidford, A.: Lower bounds for finding stationary points I. Math. Program. **184**, 71–120 (2020). https://doi.org/10.1007/s10107-019-01406-y
8. Chewi, S., Bubeck, S., Salim, A.: On the complexity of finding stationary points of smooth functions in one dimension. In Proceedings of the 34th International Conference on Algorithmic Learning Theory (ALT), 358–374, (2023). URL https://proceedings.mlr.press/v201/chewi23a.html
9. Daskalakis, C., Papadimitriou, C.: Continuous local search. In Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA), 790–804, (2011). https://doi.org/10.1137/1.9781611973082.62
10. Daskalakis, C., Skoulakis, S. and Zampetakis, M.: The complexity of constrained min-max optimization. In Proceedings of the 53rd ACM Symposium on Theory of Computing (STOC), 1466–1478, (2021). https://doi.org/10.1145/3406325.3451125
11. Fearnley, J., Goldberg, P., Hollender, A., Savani, R.: The complexity of gradient descent: CLS = PPAD ∩ PLS. J. ACM **70**(1), 1–74 (2022). https://doi.org/10.1145/3568163
12. Göös, M., Hollender, A., Jain, S., Maystre, G., Pires, W., Robere, R., Tao, R.: Separations in proof complexity and TFNP. In Proceedings of the 63rd Symposium on Foundations of Computer Science (FOCS), 1150–1161, (2022). https://doi.org/10.1109/focs54457.2022.00111
13. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: How easy is local search? J. Comput. Syst. Sci. **37**(1), 79–100 (1988). https://doi.org/10.1016/0022-0000(88)90046-3
14. Morioka, T.: Classification of search problems and their definability in bounded arithmetic. Master's thesis, University of Toronto, (2001). URL https://www.collectionscanada.ca/obj/s4/f2/dsk3/ftp04/MQ58775.pdf
15. Nemirovskij, A.S., Yudin, D.B.: Problem complexity and method efficiency in optimization. Wiley-Interscience, (1983)

16. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course. Springer Science & Business Media (2003)
17. Russell, W.S.: Polynomial interpolation schemes for internal derivative distributions on structured grids. Appl. Numer. Math. **17**(2), 129–171 (1995). https://doi.org/10.1016/0168-9274(95)00014-L
18. Vavasis, S.A.: Black-box complexity of local minimization. SIAM J. Optim. **3**(1), 60–80 (1993). https://doi.org/10.1137/0803004