The Query Complexity of Local Search and Brouwer in Rounds

Simina Brânzei SIMINA.BRANZEI@GMAIL.COM
Purdue University, USA

Jiawei Li DAVIDLEE@CS.UTEXAS.EDU
University of Texas at Austin, USA

Editors: Po-Ling Loh and Maxim Raginsky

Abstract

We consider the query complexity of finding a local minimum of a function defined on a graph, where at most k rounds of interaction (aka adaptivity) with the oracle are allowed. Adaptivity is a fundamental concept studied due to the need to parallelize computation and understand the speedups attainable. The query complexity of local search is tightly related to the complexity of computing stationary points of a function, thus bounds for local search can give insights into the performance of algorithms such as gradient descent.

We focus on the d-dimensional grid $\{1, 2, \ldots, n\}^d$, where the dimension $d \geq 2$ is a constant. Our main contribution is to give algorithms and lower bounds that characterize the trade-off between the number of rounds of adaptivity and the query complexity of local search, when the number of rounds is constant and polynomial in n, respectively.

The local search analysis also enables us to characterize the query complexity of computing a Brouwer fixed point in rounds. Our proof technique for lower bounding the query complexity in rounds may be of independent interest as an alternative to the classical relational adversary method of Aaronson from the fully adaptive setting. [1]

Keywords: local search, rounds of adaptivity, query complexity, Brouwer fixed point, parallel computation

1. Introduction

Local search is a powerful heuristic embedded in many natural processes, which is often used to solve hard optimization problems. Algorithms based on local search include gradient methods, the Lin-Kernighan algorithm for the traveling salesman problem, and the Metropolis-Hastings algorithm for sampling. Johnson et al. [1988] studied the computational complexity of local search by introducing the class PLS, which captures local search problems for which local optimality can be verified in polynomial time. There are many natural PLS-complete problems, such as finding a pure Nash equilibrium in a congestion game (Fabrikant et al. [2004]) and a locally optimum maximum cut in a graph (Schäffer and Yannakakis [1991]).

In the query complexity model for local search, we are given a graph $G = (V, E)$ and oracle access to a function $f : V \to \mathbb{R}$. The set V can represent any universe of elements with a notion of

1. The full version of the paper is at https://arxiv.org/abs/2101.00061.

© 2022 S. Brânzei & J. Li.

neighbourhood and the goal is to find a vertex v that is a local minimum: $f(v) \leq f(u)$ for all $(u, v) \in E$. The query complexity is the number of oracle queries needed to find a local minimum in the worst case.

The query complexity of local search is interesting both theoretically and due to applications such as understanding the performance of gradient methods. For example, gradient descent computes a stationary point of a function, thus the runtime of the algorithm is lower bounded by the complexity of computing a stationary point. Moreover, lower bounds for the query complexity of stationary points are often inherited from lower bounds on the query complexity of local search; the constructions have similar structure, such as embedding in the graph a random path that is hard to predict, with the local minimum or stationary point at the end of the path (see, e.g., Aldous [1983]; Vavasis [1993]; Bubeck and Mikulincer [2020]). Finally, upper bounds on the query complexity of local search can imply bounds for finding pure Nash equilibria in games such as potential games. In potential games the graph is implicitly defined, such that each node represents a strategy profile and two nodes are adjacent if an agent can move from one strategy profile to another one by changing its own strategy. A sequence of best response moves always leads to a pure Nash equilibrium, which is a local maximum of the potential function associated with the game (see, e.g., Rosenthal [1973]; Monderer and Shapley [1996]; Babichenko et al. [2019]).

The known protocols for local search with low total query complexity, such as steepest descent with a warm start (Aldous [1983]), are highly sequential, i.e. have many rounds of interaction with the function oracle. However, multiple rounds can be expensive in applications. For example, when algorithms such as gradient descent are run on data stored in the cloud, there can be delays due to back and forth messaging across the network. A remedy for such delays is designing protocols with fewer rounds, which is the focus of our work.

We consider the query complexity of local search on the d-dimensional grid in k rounds, aiming to understand the tradeoff between the number of rounds and the total query complexity. The grid is obtained by discretizing the continuous space, thus understanding it can be useful for tasks such as computing an approximate stationary point or fixed point of a function defined on the d-dimensional Euclidean space. When there are k rounds of adaptivity, an algorithm asks a set of simultaneous queries in each round i, then receives the answers, after which it issues the set of queries for round $i+1$. The algorithm stops and outputs an answer by the end of round k. This setting models parallel computation, since each simultaneous query from a given round can be sent to a different processor.

The parallel complexity of optimization was first considered by Nemirovski [1994]. This was analyzed in a series of follow-up works for submodular functions (see, e.g., Balkanski and Singer [2018]; Balkanski et al. [2019]; Ene and Nguyen [2019]; Balkanski and Singer [2020]). Algorithms with few rounds of interaction with the function oracle (aka low depth) for the problem of computing stationary points were considered in Bubeck and Mikulincer [2020]. Bubeck et al. [2019] studied parallel convex optimization where the oracle can answer $poly(d)$ queries in each round, where d is the dimension. Since the d-dimensional grid is obtained by discretizing $\mathbb{R}^d$, understanding local search on the d-dimensional grid gives insight into the hardness of computing stationary points in parallel and thus parallel implementations of gradient descent. The constant dimension model is important in the study of TFNP in complexity theory. For example, the complexity class CLS was first defined as any problem that could be reduced to the 3D-CONTINUOUS-LOCALOPT problem (Daskalakis and Papadimitriou [2011]). Göös et al. [2022] showed that EOPL = PPAD ∩ PLS using the 2D grid as a unified model to define problems from several complexity classes (PPAD, PPADS, PLS, EOPL, and SOPL).

Parallel complexity is a fundamental concept, which was studied extensively for problems such as sorting, selection, finding the maximum element of a vector, and the sorted top-k elements (Valiant [1975]; Pippenger [1987]; Bollobás [1988]; Alon et al. [1986]; Wigderson and Zuckerman [1999]; Gasarch et al. [2003b]; Braverman et al. [2016, 2019]; Cohen-Addad et al. [2020]). An overview on parallel sorting algorithms is given in the book of Akl [2014]. Babichenko et al. [2019] analyzed the communication complexity of local search, which captures the hardness of finding a local optimum in distributed environments, where data may be stored on different computers, from the point of view of the communication cost.

Our results also imply bounds for the computational problem BROUWER, where the goal is to compute an approximate fixed point of a Lipschitz function defined on the d-dimensional cube, the existence of which is guaranteed by Brouwer's theorem. BROUWER is computationally equivalent to problems such as finding a Nash equilibrium in an n-player game (Nash [1950]; Papadimitriou [1994]; Daskalakis et al. [2009]; Chen et al. [2009]) and a local min-max equilibrium in a two player game with nonconvex-nonconcave utilities (Daskalakis et al. [2021]), the latter of which has applications to training generative adversarial networks (Goodfellow et al. [2014]; Arjovsky et al. [2017]; Daskalakis et al. [2021]).

2. Model

We first introduce the model for local search and Brouwer, then define the query complexity.

Local Search Let $G = (V, E)$ be an undirected graph and $f : V \to \mathbb{R}$ a function, where $f(\mathbf{x})$ is the value of node $\mathbf{x} \in V$. We have oracle access to f and the goal is to find a local minimum, that is, a vertex $\mathbf{x}$ with the property that $f(\mathbf{x}) \leq f(\mathbf{y})$ for all neighbours $\mathbf{y}$ of $\mathbf{x}$.

We focus on the setting where the graph is a d-dimensional grid of side length n. Thus $V = [n]^d$, where $[n] = \{1, 2, \ldots, n\}$ and $(\mathbf{x}, \mathbf{y}) \in E$ if $\|\mathbf{x} - \mathbf{y}\|_1 = 1$. The dimension d for both local search and Brouwer fixed-point is a constant. Unless otherwise specified, we have $d \geq 2$.

The local search results imply bounds for the problem of finding a Brouwer fixed point, which is defined next.

Brouwer In the Brouwer setting, we are given an L-Lipschitz function $F : [0,1]^d \to [0,1]^d$, where $L > 1$ is a constant[2] such that $\|F(\mathbf{x}) - F(\mathbf{y})\|_\infty \leq L\|\mathbf{x} - \mathbf{y}\|_\infty, \forall \mathbf{x}, \mathbf{y} \in [0,1]^d$.

The computational problem is: given a tuple (ϵ, L, d, F), find a point $\mathbf{x}^* \in [0,1]^d$ such that $\|F(\mathbf{x}^*) - \mathbf{x}^*\|_\infty \leq \epsilon$. An exact fixed point exists by Brouwer's fixed point theorem.

Query complexity and rounds We have oracle access to the function f and at most k rounds of interaction with the oracle. An algorithm running in k rounds will submit in each round j a number of parallel queries, then wait for the answers, and then submit the queries for round $j+1$. The choice of queries submitted in round j can only depend on the results of queries from earlier rounds. At the end of the k-th round, the algorithm must stop and output a solution.

2. When $L < 1$ we obtain the Banach fixed-point theorem, where the unique fixed point can be approximated quickly.

The deterministic query complexity is the total number of queries necessary and sufficient to find a solution.

The randomized query complexity is the expected number of queries required to find a solution with probability at least $2/3$ for any input, where the expectation is taken over the coin tosses of the protocol.[3]

3. Our Results

Local Search

We show the following bounds for local search on the d-dimensional grid $[n]^d$ in k rounds.

Theorem 1 *(Local search, constant rounds) Let $d, k \in \mathbb{N}$ be constant, where $d \geq 2$ and $k \geq 1$. The query complexity of local search in k rounds on the d-dimensional grid $[n]^d$ is $\Theta\big(n^{\frac{d^{k+1}-d^k}{d^k-1}}\big)$, for both deterministic and randomized algorithms.*

For example, when $k = 2$, the query complexity is $\Theta\big(n^{\frac{d^2}{d+1}}\big)$ for both deterministic and randomized algorithms. When $k \to \infty$, the bound of Theorem 1 is close to $\Theta(n^{d-1})$, with gap smaller than any polynomial. The classical result in Llewellyn et al. [1993] showed that the query complexity of local search for deterministic fully adaptive algorithms is $\Theta(n^{d-1})$, and the upper bound is achieved by a divide-and-conquer algorithm with $O(\log n)$ rounds.

Our result fills the gap between one round algorithms and logarithmic rounds algorithms except for a small margin. This theorem also implies that randomness does not help when the number of rounds is constant.

Theorem 2 *(Local search, polynomial rounds) Consider the d-dimensional grid with side length $n \in \mathbb{N}$, where $d \geq 2$. Let $k = n^\alpha \in \mathbb{N}$, where $\alpha \in (0, d/2)$ is a constant. The randomized query complexity of local search in k rounds on the d-dimensional grid $[n]^d$ is:*

- $\Theta\left(n^{(d-1)-\frac{d-2}{d}\alpha}\right)$ *when $d \geq 5$;*
- $O\left(n^{3-\frac{\alpha}{2}}\right)$ *and* $\widetilde{\Omega}\left(n^{3-\frac{\alpha}{2}}\right)$ *when $d = 4$;*
- $O\left(n^{2-\frac{\alpha}{3}}\right)$ *and* $\Omega\left(\max(n^{2-\frac{2\alpha}{3}}, n^{\frac{3}{2}})\right)$ *when $d = 3$.*

When $\alpha \to 0$, the bound approaches $\Theta(n^{d-1})$, i.e., the bound of constant and logarithmic rounds algorithm. When $\alpha \to (d/2)$, the upper bound is close to $\Theta(n^{\frac{d}{2}})$, i.e., the fully adaptive algorithm. Thus, our result fills the gaps between constant (or logarithmic) rounds algorithms and fully adaptive algorithms, except for a small gap when $d \in \{3, 4\}$. The bound for $d = 2$ in polynomial number of rounds was known (Sun and Yao [2009]; Llewellyn et al. [1993]).

For $d = 1$, we show that the query complexity of computing a local minimum on the 1-dimensional grid $[n]$ in k rounds is $\Theta\left(n^{1/k}\right)$, for both deterministic and randomized algorithms.

3. Any other constant greater than $1/2$ will suffice.

A summary of our results for local search on the d-dimensional grid can be found in Table 1, together with the bounds known in the existing literature.

Local search on the grid $[n]^d$	Deterministic	Randomized
Constant rounds: $k = O(1)$	$\Theta\big(n^{\frac{d^{k+1}-d^k}{d^k-1}}\big)$ (*)	$\Theta\big(n^{\frac{d^{k+1}-d^k}{d^k-1}}\big)$ (*)
Polynomial rounds: $k = n^\alpha$, $\alpha \in (0, d/2)$	$\Theta\left(n^{d-1}\right)$ (Llewellyn et al.; Llewellyn and Tovey; Althöfer and Koschnick)	$d \geq 5$: $\Theta\left(n^{(d-1)-\frac{d-2}{d}\alpha}\right)$ (*)
		$d = 4$: $O\left(n^{3-\frac{\alpha}{2}}\right)$ and $\widetilde{\Omega}\left(n^{3-\frac{\alpha}{2}}\right)$ (*)
		$d = 3$: $O\left(n^{2-\frac{\alpha}{3}}\right)$ and $\Omega\left(\max(n^{2-\frac{2\alpha}{3}}, n^{\frac{3}{2}})\right)$ (*)
		$d = 2$: $\widetilde{\Theta}(n)$ (Sun and Yao; Llewellyn et al.)
Fully adaptive: $k = \infty$	$\Theta\left(n^{d-1}\right)$ (Llewellyn et al.; Llewellyn and Tovey; Althöfer and Koschnick)	$d \geq 3$: $\widetilde{\Theta}\big(n^{\frac{d}{2}}\big)$ (Aldous; Zhang) $d = 2$: $\widetilde{\Theta}(n)$ (Sun and Yao; Llewellyn et al.)

Table 1: Query complexity of local search in k rounds on d-dimensional grid of side length n, for $d \geq 2$. Our results are marked with (*). The deterministic divide-and-conquer algorithm Llewellyn et al. [1993] takes $O(\log n)$ rounds, while the randomized warm-start algorithm Aldous [1983] needs $O\big(n^{\frac{d}{2}}\big)$ rounds. For deterministic fully adaptive algorithms, the algorithm is given by Llewellyn et al. [1993], while the lower bound is from Llewellyn and Tovey [1993] and Althöfer and Koschnick [1993].

At a high level, when the number of rounds k is constant, the optimal algorithm is closer to the deterministic divide-and-conquer algorithm for local search designed by Llewellyn et al. [1993]. When the number of rounds is polynomial (i.e. $k = n^\alpha$, for $0 < \alpha < d/2$), the algorithm is closer to the randomized algorithm from the fully adaptive setting, which does steepest descent with a warm start (Aldous [1983]).

Brouwer

Our local search results above also imply a characterization for the query complexity of finding an approximate Brouwer fixed point in constant number of rounds on the d-dimensional cube. For Brouwer we consider only constant rounds, since Brouwer can be solved optimally in $O(\log(1/\epsilon))$ rounds (Chen and Deng [2005]).

Theorem 3 *[Brouwer, constant rounds] Let $d, k \in \mathbb{N}$ be constant. For any $\epsilon > 0$, the query complexity of the ϵ-approximate Brouwer fixed-point problem in the d-dimensional unit cube $[0,1]^d$ with k rounds is $\Theta\big((1/\epsilon)^{\frac{d^{k+1}-d^k}{d^k-1}}\big)$, for both deterministic and randomized algorithms.*

We also show that when $d = 1$, the query complexity of finding an ϵ-approximate Brouwer fixed point in k rounds is $\Theta\left((1/\epsilon)^{1/k}\right)$, for both deterministic and randomized algorithms.

3.1. Roadmap of the paper

Section 4 discusses related work. Section 5 overviews the results and proof techniques for local search. Section 6 overviews the results and proofs for Brouwer.

4. Related Work

The query complexity of local search was studied first experimentally by Tovey [1981], while Aldous [1983] gave the first theoretical analysis. For any graph G with n vertices and maximum degree d, Aldous' algorithm can be seen as steepest descent with warm start and works as follows: first query t vertices $x_1, \ldots, x_t$ selected uniformly at random and pick the vertex x^* that minimizes the function among these [4]. Then run steepest descent from x^* and stop when no further improvement can be made, returning the final vertex reached. When $t = \sqrt{nd}$, the algorithm issues $O(\sqrt{nd})$ queries in expectation and has roughly as many rounds of interaction with the oracle.

For the n-dimensional hypercube, Aldous [1983] gave an upper bound of $O(n \cdot 2^{n/2})$ and a lower bound of $\Omega(2^{n/2-o(n)})$ for randomized algorithms. The lower bound construction from Aldous [1983] is as follows. Consider an initial vertex v_0 uniformly at random. The function value at v_0 is $f(v_0) = 0$. From this vertex, start an unbiased random walk $v_0, v_1, \ldots$ For each vertex v in the graph, set $f(v)$ equal to the first hitting time of the walk at v; that is, $f(v) = \min\{t \mid v_t = v\}$. The function f defined this way has a unique local minimum at $f(v_0)$. By analyzing this distribution, Aldous [1983] showed a lower bound of $2^{n/2-o(n)}$ on the hypercube.

Aldous' algorithm described above is essentially optimal for graphs such as the hypercube and the d-dimensional grid (Aldous [1983]; Sun and Yao [2009]; Zhang [2009]). At the same time, the algorithm is highly sequential. Llewellyn, Tovey, and Trick [1993] considered the deterministic query complexity of local search and devised a divide-and-conquer approach, which has higher total query complexity but uses fewer rounds. Their algorithm is deterministic and identifies in the first step a vertex separator S of the input graph G [5]. Afterwards, it queries all the vertices in S to find the minimum v among these. If v is a local minimum of G, then return it. Otherwise, there is a neighbour w of v with $f(w) < f(v)$. Repeat the whole procedure on the new graph G', defined as the connected component of $G \setminus S$ containing w. Correctness holds since the steepest descent from w cannot escape G'. On the d-dimensional grid, the vertex separator S can be defined as the $(d-1)$-dimensional wall that divides the current connected component evenly; thus a local optimum can be found with $O(n^{d-1})$ queries in $O(\log n)$ rounds.

Llewellyn and Tovey [1993]; Althöfer and Koschnick [1993] applied the adversarial argument proposed in Llewellyn et al. [1993] to show that $\Omega(n^{d-1})$ queries are necessary for any deterministic algorithm on the d-dimensional grid of side length n. Llewellyn et al. [1993] also studied arbitrary graphs, showing that $O(\sqrt{n} + \delta \log n)$ queries are sufficient on graphs with n vertices when the maximum degree of the graph is δ and the graph has constant genus.

We observe the contrast between the randomized algorithm by Aldous [1983], which is almost sequential, running in $O(n^{d/2})$ rounds, and the deterministic divide-and-conquer algorithm by

4. That is, the vertex x^* is defined as: $x^* = x_j$, where $j = \arg\min_{i=1}^{t} f(x_i)$.
5. A vertex separator is a set of vertices $S \subseteq V$ with the property that there exist vertices $u, v \in V$, where V is the set of vertices of G, such that any path between u and v passes through S.

Llewellyn et al. [1993], which can be implemented in $O(\log n)$ rounds. Even though the randomized algorithm (Aldous [1983]) is (essentially) optimal in terms of number of queries, it takes many rounds. Thus it is natural to ask whether this algorithm can be parallelized and what is the tradeoff between the total query complexity and the number of rounds.

Aaronson [2006] improved the bounds given by Aldous [1983] for randomized algorithms by designing a novel technique called the *relational adversarial method* inspired by the adversarial method in quantum computing. This method avoids analyzing the posterior distribution during the execution directly and gave improved lower bounds for both the hypercube and the grid. Follow-up work by Zhang [2009] and Sun and Yao [2009] obtained even tighter lower bounds for the grid using this method with better choices on the random process; their lower bound is $\widetilde{\Omega}\left(n^{\frac{d}{2}}\right)$, which is nearly optimal.

The computational complexity of local search is captured by the class PLS, which was defined by Johnson, Papadimitriou, and Yannakakis [1988] to model the difficulty of finding locally optimal solutions to optimization problems. A class related to PLS is PPAD, introduced by Papadimitriou [1994] to study the computational complexity of finding a Brouwer fixed-point. PPAD contains many natural problems that are computationally equivalent to the problem of finding a Brouwer fixed point (Chen and Deng [2009]), such as finding an approximate Nash equilibrium in a multiplayer or two-player game (Daskalakis et al. [2009]; Chen et al. [2009]), an Arrow-Debreu equilibrium in a market (Vazirani and Yannakakis [2011]; Chen et al. [2017]), and a local min-max point (Daskalakis, Skoulakis, and Zampetakis [2021]). The query complexity of computing an ϵ-approximate Brouwer fixed point was studied in a series of papers for fully adaptive algorithms starting with Hirsch, Papadimitriou, and Vavasis [1989], later improved by Chen and Deng [2005] and Chen and Teng [2007].

The classes PLS and PPAD are related, both being a subset of TFNP. Fearnley, Goldberg, Hollender, and Savani [2021] showed that the class CLS, introduced by Daskalakis and Papadimitriou [2011] to capture continuous local search, is equal to PPAD $\cap$ PLS. The query complexity of continuous local search has also been studied (see, e.g., Hubáček and Yogev [2017]).

Valiant [1975] initiated the study of parallelism using the number of comparisons as a complexity measure and showed that p processor parallelism can offer speedups of at least $O\left(\frac{p}{\log \log p}\right)$ for problems such as sorting and finding the maximum of a list of $n > p$ elements. Nemirovski [1994] studied the parallel complexity of optimization, with many recent results on the tradeoff between the rounds of adaptivity and the total query complexity in submodular optimization (see, e.g. Balkanski and Singer [2018]; Balkanski et al. [2019]; Ene and Nguyen [2019]; Balkanski and Singer [2020]). An overview on parallel sorting algorithms is given in the book by Akl [2014] and many works on sorting and selection in rounds can be found in Valiant [1975]; Pippenger [1987]; Bollobás [1988]; Alon et al. [1986]; Wigderson and Zuckerman [1999]; Gasarch et al. [2003a], aiming to understand the tradeoffs between the number of rounds of interaction and the query complexity.

Another setting of interest where rounds are important is active learning, where there is an "active" learner that can submit queries—taking the form of unlabeled instances—to be annotated by an oracle (e.g., a human) (Settles [2012]). However each round of interaction with the human annotator has a cost, which can be captured through a budget on the number of rounds.

5. Local Search

In this section we state our results for local search and give an overview of the proofs.

5.1. Local Search in Constant Rounds

When the number of rounds k is a constant, we obtain the following bounds.

Theorem 1 (Local search, constant rounds, restated): *Let $d, k \in \mathbb{N}$ be constant, where $d \geq 2$ and $k \geq 1$. The query complexity of local search in k rounds on the d-dimensional grid $[n]^d$ is $\Theta\left(n^{\frac{d^{k+1}-d^k}{d^k-1}}\right)$, for both deterministic and randomized algorithms.*

For example, when $k = 2$, the query complexity is $\Theta\left(n^{\frac{d^2}{d+1}}\right)$.

5.1.1. Upper bound overview for local search in constant rounds.

The algorithm for constant rounds works as follows. Initialize a cube C_0 to the whole grid $[n]^d$.

In each round $i = 1, \ldots, k-1$, the algorithm divides the current cube C_{i-1} into a set of mutually exclusive sub-cubes $C_i^1, \ldots, C_i^{n_i}$ of side length ℓ_i (for a carefully set value of ℓ_i) that cover C_{i-1}. Then it queries all the points on the boundary of the sub-cubes $C_i^1, \ldots, C_i^{n_i}$ and select the point $\mathbf{x}_i^*$ with minimal value among them. Set $C_i = C_i^j$, where C_i^j is the sub-cube that $\mathbf{x}_i^*$ belongs to and repeat. Finally, in round k, query all the points in the current cube C_{k-1} and find the solution point.

Figure 1 shows an illustration for the case $k = 2$ and $d = 2$.

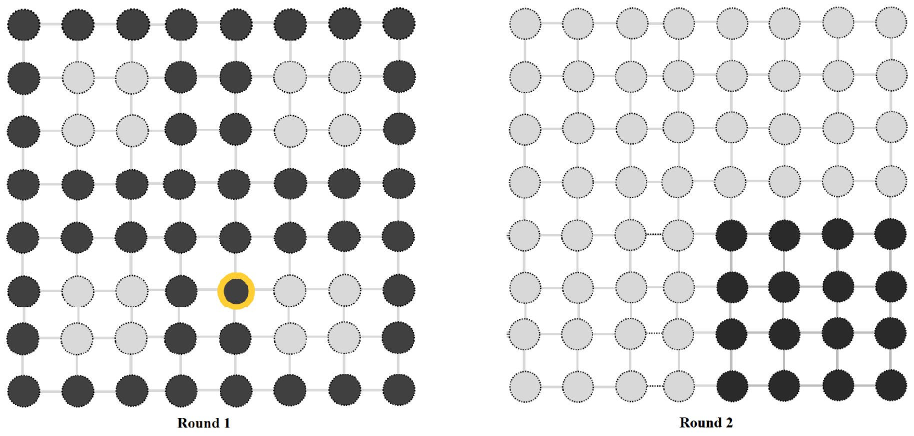


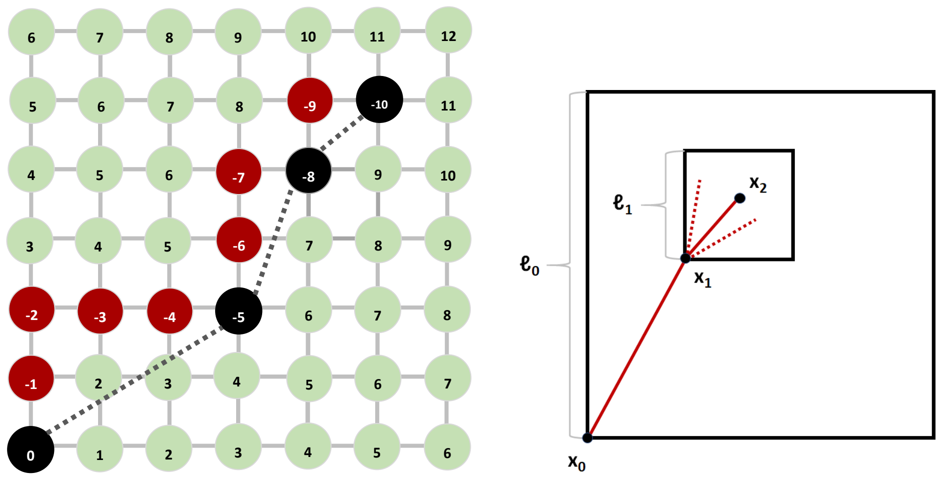
Figure 1: 2D grid of size 8×8. Suppose there are two rounds. In round 1, illustrated on the left, the algorithm queries all the black points and selects the minimum among all these points (shown in yellow). In round 2, illustrated on the right, it queries the entire sub-square in which the minimum from the first round was found. When $d = 2$ and $k = 2$, the query complexity is $\Theta\left(n^{\frac{4}{3}}\right)$ by setting $\ell_1 := n^{\frac{2}{3}}$.

The main observation that allows the proof to work is that when the space is divided into subcubes and $\mathbf{x}_i^*$ is the point with smallest value on the boundaries of all the sub-cubes, steepest descent started at $\mathbf{x}_i^*$ cannot escape the sub-cube it is contained in. By choosing the side lengths of the sub-cubes appropriately, we get the required upper bound. The detailed proof is included in the full version of the paper, together with the other omitted proofs.

5.1.2. LOWER BOUND OVERVIEW FOR LOCAL SEARCH IN CONSTANT ROUNDS.

To show lower bounds, we apply Yao's minimax theorem (Yao [1977]): first we provide a hard distribution of inputs, then show that no *deterministic* algorithm could achieve accuracy larger than some constant on this distribution. The hard distribution will be given by a *staircase* construction (Vavasis [1993]; Hirsch et al. [1989]). A staircase will be a random path with the property that the unique local optimum is hidden at the end of the path. We present a sketch here; see the full version of the paper for the complete calculations.

Figure 2 shows an example of a staircase, which consists of the black and red vertices. The bottom left black vertex is the starting point of the staircase and the value of the function there is set to zero. Then the value decreases by one with each step along the staircase, like going down the stairs. The value of the function at any point outside the staircase is equal to the distance to the entrance of the staircase.



(*a*) Staircase in 2D for 3 rounds. (*b*) Stylized staircase in 2D.

Figure 2: Figure (a) shows a two dimensional staircase for $k = 3$ rounds. The number of black points is equal to $k + 1$. The black points are connected by "folded segments" shown in red. The idea is that in each round, the algorithm will learn at most one more folded segment. Figure (b) shows in a stylized way the process of selecting the next part of the staircase given that the first folded segment was determined. The folded segments are displayed here as straight lines for simplicity.

Intuitively, the algorithm cannot find much useful structural information of the input and thus has no advantage over a path following algorithm. The staircase construction can be embedded in two different tasks: finding a local minimum of a function (Aldous [1983]; Aaronson [2006]; Sun and

Yao [2009]; Zhang [2009]; Hubáček and Yogev [2017]) and computing a Brouwer fixed-point (Chen and Teng [2007]; Hirsch et al. [1989]).

The most challenging part is rigorously proving that such intuition is correct. The lower bound proof from Aldous [1983] for the hypercube $\{0,1\}^n$ has complex probability analysis on the random walk and relies on the structural property of hypercube $\{0,1\}^n$.

All other works on the randomized lower bound of local search are based on Aaronson's relational adversarial method (Aaronson [2006]). However, we found it's inherently difficult to consider multiple queries in one round in the framework of relational adversarial method. Therefore, our main technical innovation is a new technique to incorporate the round limit into the randomized lower bounds.This could also serve as a simpler alternative method of the classical relational adversarial method (Aaronson [2006]) in the fully adaptive setting.

Staircase definition. We define a staircase as an array of connecting grid points $(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t)$, for $0 \leq t \leq k$. A uniquely determined path called folded segment is used to link every two consecutive points $(\mathbf{x}_i, \mathbf{x}_{i+1})$. The start point $\mathbf{x}_0$ is fixed at corner $\mathbf{1}$, and the remaining connecting points are chosen randomly in a smaller and smaller cube region with previous connecting points as corner. For k round algorithms, we choose a distribution of staircases of "length" k, where the length is defined as the number of connecting points in the staircase minus 1.

Good staircases. We say that a length t staircase is "good" with respect to a deterministic algorithm $\mathcal{A}$ if for each $1 \leq i < t$, any point in the suffix of the staircase (i.e. after the connecting point $\mathbf{x}_i$ [6]) is not queried in rounds $1, \ldots, i$, when $\mathcal{A}$ runs on the input generated by this staircase.

The input functions generated by good staircases are like adversarial inputs: $\mathcal{A}$ could only (roughly) learn the location of the next connecting point $\mathbf{x}_i$ in each round i, and still know little about the staircase from $\mathbf{x}_i$ onwards.

We show that if $9/10$ of all possible length k staircases are good, then the algorithm will make a mistake with probability at least $7/40$. We ensure that each possible staircase is chosen with the same probability; their total number is easy to estimate.

Thus the main technical part of our proof is counting the number of good staircases.

Counting good staircases. We show the next properties about the *prefix* of good staircases:

P1: If s is a good staircase, then any "prefix" s' of s is also a good staircase.

P2: Let s_1, s_2 be two good staircases with respect to algorithm $\mathcal{A}$. If the first $i+1$ connecting points of the staircases are same, then $\mathcal{A}$ will submit the same queries in rounds $1, \ldots, i+1$ when given as input the functions generated by s_1 and s_2, respectively.

We first fix a good staircase $s^{(i-1)}$ of length $i-1$ and consider two good staircases s_1, s_2 of length i that have $s^{(i-1)}$ as prefix. By P2, the algorithm $\mathcal{A}$ will make the *same* queries in rounds $1, \ldots, i+1$ when running on the inputs generated by s_1 and s_2, respectively. This enables estimating the total number of good length $i+1$ staircase with $s^{(i-1)}$ as prefix.

6. That is, the point $\mathbf{x}_i$ is not included.

By summing over all good staircases of length $i-1$, we get a recursive equation between the number of good staircases of length $i-1$, i, and $i+1$. This will be used to show that most staircases of length k are good.

5.2. Local Search in Polynomial Rounds

When the number of rounds k is polynomial in n, that is $k = n^{\alpha}$ for some constant $\alpha > 0$, the algorithm that yields the upper bound in Theorem 1 is no longer efficient.

We design a different algorithm for this regime and also show an almost matching lower bound. With polynomial rounds we can focus on $d \geq 3$. Sun and Yao [2009] proved a lower bound of $\widetilde{\Omega}(n)$ for fully adaptive algorithms in 2D and the divide-and-conquer algorithm by Llewellyn et al. [1993] achieves this bound with only $O(\log(n))$ rounds.

Theorem 2 (Local search, polynomial rounds, restated): *Consider the d-dimensional grid with side length $n \in \mathbb{N}$, where $d \geq 2$. Let $k = n^{\alpha} \in \mathbb{N}$, where $\alpha \in (0, d/2)$ is a constant. The randomized query complexity of local search in k rounds on the d-dimensional grid $[n]^d$ is:*

- $\Theta\left(n^{(d-1)-\frac{d-2}{d}\alpha}\right)$ *when $d \geq 5$;*
- $O\left(n^{3-\frac{\alpha}{2}}\right)$ *and* $\widetilde{\Omega}\left(n^{3-\frac{\alpha}{2}}\right)$ *when $d = 4$;*
- $O\left(n^{2-\frac{\alpha}{3}}\right)$ *and* $\Omega\left(\max(n^{2-\frac{2\alpha}{3}}, n^{\frac{3}{2}})\right)$ *when $d = 3$.*

5.2.1. UPPER BOUND OVERVIEW FOR LOCAL SEARCH IN POLYNOMIAL ROUNDS.

Since the constant rounds algorithm is not optimal in this regime, we design an algorithm that randomly samples many points in round 1 and then starts searching for the solution from the smallest valued point from round 1. This is similar to the algorithm in Aldous [1983], except the steepest descent part of Aldous' algorithm is highly sequential.

To get better parallelism, we design a recursive procedure ("fractal-like steepest descent") which parallelizes the steepest descent steps at the cost of more queries. We present the main ideas next:

Sequential procedure. Let $C(\mathbf{x}, s) := \{\mathbf{y} \in [n]^d : \|\mathbf{y}-\mathbf{x}\|_\infty \leq s\}$ be the set of grid points in the d-dimensional cube of side length $2 \cdot s$, centered at point $\mathbf{x}$. Let $\text{rank}(\mathbf{x})$ be the number of points with smaller function value than point $\mathbf{x}$.

Assume we already have a procedure P and a number $s < n$ such that $P(\mathbf{x})$ will either return a point $\mathbf{y} \in C(\mathbf{x}, s)$ with $\text{rank}(\mathbf{y}) \leq \text{rank}(\mathbf{x}) - s$, or output a correct solution and halt. Suppose in both cases $P(\mathbf{x})$ takes at most r rounds and Q queries in total for any $\mathbf{x}$.

If we want to find a point $\mathbf{x}^*$ with $\text{rank}(\mathbf{x}^*) \leq \text{rank}(\mathbf{x}_0) - t \cdot s$ for any given $\mathbf{x}_0$ or output a correct solution, the naive approach is to run P *sequentially* t times, taking

$$\mathbf{y}_1 = P(\mathbf{x}_0), \mathbf{y}_2 = P(\mathbf{y}_1), \ldots, \mathbf{x}^* = \mathbf{y}_t = P(\mathbf{y}_{t-1}).$$

Since each call of P must wait for the result from the previous call, the naive approach will take $t \cdot r$ rounds and $t \cdot Q$ queries.

Parallel procedure. We can parallelize the previous procedure P using auxiliary variables that are more expensive in queries, but cheap in rounds. For $i \in [t]$, let $\mathbf{x}_i$ be the point with minimum function value on the boundary of cube $C(\mathbf{x}_{i-1}, s)$, which can be found in only *one* round with $O(s^{d-1})$ queries after getting $\mathbf{x}_{i-1}$, i.e., we get the location of $\mathbf{x}_{i-1}$ at the start of round i. Next, we take $\mathbf{y}_i$ to be $P(\mathbf{x}_{i-1})$ instead of $P(\mathbf{y}_{i-1})$; thus the location of $\mathbf{y}_i$ will be available at round $i + r$. To ensure correctness, we will compare the value of point $\mathbf{y}_i$ with the value of point $\mathbf{x}_i$. If $f(\mathbf{x}_i) \leq f(\mathbf{y}_i)$ then

$$\operatorname{rank}(\mathbf{x}_i) \leq \operatorname{rank}(\mathbf{y}_i) \leq \operatorname{rank}(\mathbf{x}_{i-1}) - s\,. \tag{1}$$

Otherwise, since $\mathbf{y}_i \in C(\mathbf{x}_{i-1}, s)$ has smaller value than any point on the boundary of $C(\mathbf{x}_{i-1}, s)$, we could use a slightly modified version of the divide-and-conquer algorithm of Llewellyn et al. [1993] to find the solution within the sub-cube $C(\mathbf{x}, s)$ in $\log n$ rounds and $O(s^{d-1})$ queries, and then halt all running procedures. If $f(\mathbf{x}_i) \leq f(\mathbf{y}_i)$ holds for any i, applying inequality 1 for t times we will get $\operatorname{rank}(\mathbf{x}) \leq \operatorname{rank}(\mathbf{x}_t) - t \cdot s$, so we could return $\mathbf{x}^* = \mathbf{x}_t$ in this case. This parallel approach will take only $t + r$ rounds and $O(t \cdot (Q + s^{d-1}))$ queries.

The base case of procedure P is the steepest descent algorithm. Then, multiple layers of the recursive process as described above are implemented to ensure the round limit is met. The parameters of the algorithm, such as s and t above, are described in the full version.

5.2.2. Lower bound overview for local search in polynomial rounds.

For polynomial rounds, we still use a staircase construction and hide the solution at the end of the staircase. Recall the bottom left vertex will be the starting point of the staircase and the value of the function there is set to zero. Then the value decreases by one with each step along the path. The value of the function at any point outside the staircase is equal to the distance to the entrance of the staircase.

However, the case of polynomial number of rounds is both conceptually and technically more challenging. We explain the main ideas next.

Choice of random walk Let $\mathcal{Q}_k$ denote the total number of queries allowed for an algorithm that runs in k rounds. Let $\mathcal{Q} = \mathcal{Q}_k/k$ be the average number of queries in each round. The minimum point among $\mathcal{Q}_k/2$ uniformly random queries will be at most $100 \cdot n^d/\mathcal{Q}_k$ steps away from the solution with high probability.

We set the number of points in the staircase to $\Theta(n^d/\mathcal{Q}_k)$. This strikes a balance between two extremes. If the staircase is too long, then an algorithm like steepest descent with warm-start (Aldous [1983]), which starts by querying many random points in round 1, is likely to hit the staircase in a region that is $\mathrm{O}(n^d/\mathcal{Q}_k)$ close to the endpoint. If the staircase is too short, then an algorithm such as steepest descent will find the end of the staircase in a few rounds.

Since we choose the staircase via a random walk, there are two factors affect the difficulty of finding the solution: the mixing time and what we call the "local predictability" of the walk.

Consider the random walks in Figure 3. The first random walk (Figure 3, left) randomly moves to one of its neighbor in each step. This random walk has very low local predictability and may be difficult for fully adaptive algorithms to learn. However, it mixes more slowly, which could be exploited by an algorithm with multiple queries per round.

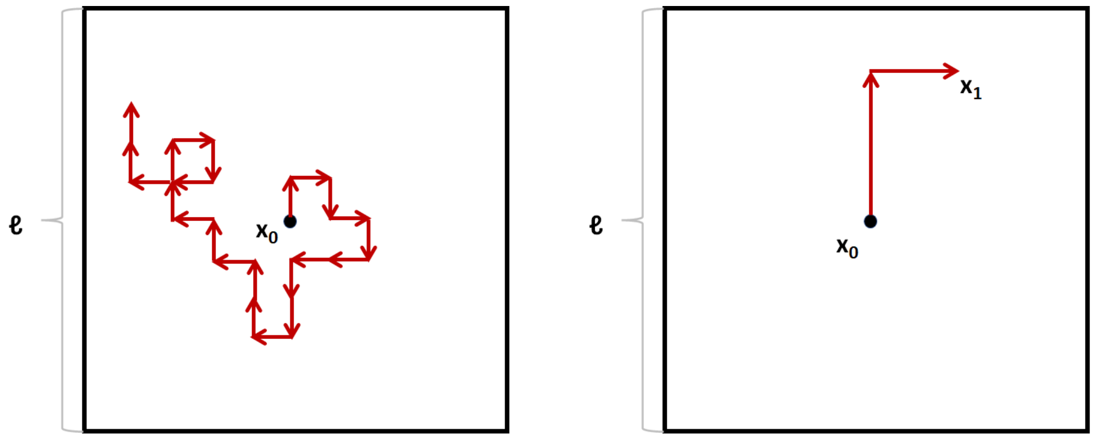


Figure 3: A 2D illustration for two types of random walk for the lower bound in polynomial rounds. The random walk on the left is more convoluted locally, but also mixes slower; the walk on the right has simpler structure, but mixes faster.

The second random walk (Figure 3, right) moves from x_0 to a uniform random point x_1 selected from a cube of side length ℓ centered at x_0 and uses d straight segments to connect these two points. This random walk is more locally predictable, since each straight segment of length $O(\ell)$ could be learned with $O(\log \ell)$ queries via binary search. Thus the walk could be learned efficiently by a fully adaptive algorithm. On the other hand, the random walk mixes faster than the one in the left figure: it takes only $O(\ell)$ points to mix in the cube region of side length ℓ. Thus, the second random walk is better when there aren't enough rounds to find each straight segment via binary search.

By controlling the parameter ℓ, we get a trade-off between the mixing time and the local predictability when the total length of the walk is fixed. We choose $\ell = \Theta(\mathcal{Q}^{1/(d-1)}) = n^{1-2\alpha/d}$ for $k = n^{\alpha}$ in our proof, which corresponds to the max possible side length of the cube if using $\mathcal{Q}$ queries to cover its boundary.

Measuring the Progress Good staircases are a central concept in the proof for constant rounds. Roughly, the algorithm can learn the location of exactly one more connecting point in each round. However, such a requirement is too strong with polynomial rounds.

Instead, we allow the algorithm to learn more than one connecting points in some rounds, while showing that it learns no more than two connecting points in each round in expectation.

Using amortized analysis, we quantify the maximum possible *progress* of an algorithm in each round by a constant Γ, which only depends on the random walk, not the algorithm. Constant Γ could be viewed as the difficulty of the random walk, which takes both the mixing time and the local predictability into account.

6. Brouwer

The problem of finding an ϵ-approximate fixed point of a continuous function was defined in Section 2. To quantify the query complexity of this problem, it is useful to consider a discrete version, obtained by discretizing the unit cube $[0,1]^d$. The discrete version of Brouwer is equivalent to the approximate fixed point problem in the continuous setting (Chen and Deng [2005]).

The algorithm for Brouwer is reminiscent of the constant rounds algorithm for local search. We divide the space in sub-cubes then find the one guaranteed to have a solution by checking a boundary condition given in Chen and Deng [2005]. Then a parity argument will show there is always a sub-cube satisfying the boundary condition. In the last round, the algorithm queries all the points in the remaining sub-cube and returns the solution.

The randomized lower bound for Brouwer is obtained by reducing local search instances generated by staircases to discrete fixed-point instances. We can naturally let the staircase within the local search instance to be the long path in discrete fixed-point problem.

7. Acknowledgements

We thank the anonymous reviewers for helpful feedback.

References

Scott Aaronson. Lower bounds for local search by quantum arguments. *SIAM Journal on Computing*, 35(4):804–824, 2006.

Selim Akl. *Parallel Sorting Algorithms*. Academic Press, 2014.

David Aldous. Minimization algorithms and random walk on the d-cube. *The Annals of Probability*, 11(2):403–413, 1983.

Noga Alon, Yossi Azar, and Uzi Vishkin. Tight complexity bounds for parallel comparison sorting. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 502–510. IEEE, 1986.

Ingo Althöfer and Klaus-Uwe Koschnick. On the deterministic complexity of searching local maxima. *Discret. Appl. Math.*, 43(2):111–113, 1993. doi: 10.1016/0166-218X(93)90002-6. URL https://doi.org/10.1016/0166-218X(93)90002-6.

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

Yakov Babichenko, Shahar Dobzinski, and Noam Nisan. The communication complexity of local search. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, page 650–661. Association for Computing Machinery, 2019.

Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, Los Angeles, CA, USA, June 25-29, 2018*, pages 1138–1151. ACM, 2018. doi: 10.1145/3188745.3188752. URL https://doi.org/10.1145/3188745.3188752.

Eric Balkanski and Yaron Singer. A lower bound for parallel submodular minimization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 130–139. ACM, 2020.

Eric Balkanski, Aviad Rubinstein, and Yaron Singer. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) 2019*, pages 283–302. SIAM, 2019.

Béla Bollobás. Sorting in rounds. *Discrete Mathematics*, 72(1-3):21–28, 1988.

Mark Braverman, Jieming Mao, and S. Matthew Weinberg. Parallel algorithms for select and partition with noisy comparisons. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 851–862. ACM, 2016. doi: 10.1145/2897518.2897642. URL https://doi.org/10.1145/2897518.2897642.

Mark Braverman, Jieming Mao, and Yuval Peres. Sorted top-k in rounds. In Alina Beygelzimer and Daniel Hsu, editors, *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pages 342–382. PMLR, 2019. URL http://proceedings.mlr.press/v99/braverman19a.html.

Sébastien Bubeck and Dan Mikulincer. How to trap a gradient flow. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 940–960. PMLR, 2020.

Sébastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, and Aaron Sidford. Complexity of highly parallel non-smooth convex optimization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13900–13909, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3c0cd9bcd0686e8bc0a9047eae120cc5-Abstract.html.

Xi Chen and Xiaotie Deng. On algorithms for discrete and approximate brouwer fixed points. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 323–330, 2005.

Xi Chen and Xiaotie Deng. On the complexity of 2d discrete fixed point problem. *Theor. Comput. Sci.*, 410(44):4448–4456, 2009. doi: 10.1016/j.tcs.2009.07.052. URL https://doi.org/10.1016/j.tcs.2009.07.052.

Xi Chen and Shang-Hua Teng. Paths beyond local search: A tight bound for randomized fixed-point computation. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 124–134. IEEE, 2007.

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *J. ACM*, 56(3):14:1–14:57, 2009. doi: 10.1145/1516512.1516516. URL https://doi.org/10.1145/1516512.1516516.

Xi Chen, Dimitris Paparas, and Mihalis Yannakakis. The complexity of non-monotone markets. *J. ACM*, 64(3):20:1–20:56, 2017. doi: 10.1145/3064810. URL https://doi.org/10.1145/3064810.

Vincent Cohen-Addad, Frederik Mallmann-Trenn, and Claire Mathieu. Instance-optimality in the noisy value-and comparison-model. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2124–2143. SIAM, 2020. doi: 10.1137/1.9781611975994.131. URL https://doi.org/10.1137/1.9781611975994.131.

Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 790–804. SIAM, 2011. doi: 10.1137/1.9781611973082.62. URL https://doi.org/10.1137/1.9781611973082.62.

Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009. doi: 10.1137/070699652. URL https://doi.org/10.1137/070699652.

Constantinos Daskalakis, Stratis Skoulakis, and Manolis Zampetakis. The complexity of constrained min-max optimization. In *Proceedings of the ACM Symposium on Theory of Computing*. Association for Computing Machinery, 2021.

Alina Ene and Huy L. Nguyen. Submodular maximization with nearly-optimal approximation and adaptivity in nearly-linear time. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, page 274–282, USA, 2019. Society for Industrial and Applied Mathematics.

Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 604–612. ACM, 2004. doi: 10.1145/1007352.1007445. URL https://doi.org/10.1145/1007352.1007445.

John Fearnley, Paul W. Goldberg, Alexandros Hollender, and Rahul Savani. The complexity of gradient descent: CLS = PPAD $\cap$ PLS. In *Proceedings of the ACM Symposium on Theory of Computing*. Association for Computing Machinery, 2021.

William Gasarch, Evan Golub, and Clyde Kruskal. Constant time parallel sorting: an empirical view. *Journal of Computer and System Sciences*, 67(1):63–91, 2003a.

William I. Gasarch, Evan Golub, and Clyde P. Kruskal. Constant time parallel sorting: an empirical view. *J. Comput. Syst. Sci.*, 67(1):63–91, 2003b. doi: 10.1016/S0022-0000(03)00040-0. URL https://doi.org/10.1016/S0022-0000(03)00040-0.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in tfnp, 2022. URL https://arxiv.org/abs/2202.07761.

Michael D Hirsch, Christos H Papadimitriou, and Stephen A Vavasis. Exponential lower bounds for finding brouwer fix points. *Journal of Complexity*, 5(4):379–416, 1989.

Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1352–1371. SIAM, 2017.

David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988. doi: 10.1016/0022-0000(88)90046-3. URL https://doi.org/10.1016/0022-0000(88)90046-3.

Donna Crystal Llewellyn and Craig A. Tovey. Dividing and conquering the square. *Discret. Appl. Math.*, 43(2):131–153, 1993. doi: 10.1016/0166-218X(93)90004-8. URL https://doi.org/10.1016/0166-218X(93)90004-8.

Donna Crystel Llewellyn, Craig Tovey, and Michael Trick. Local optimization on graphs: Discrete applied mathematics 23 (1989) 157–178. *Discrete Applied Mathematics*, 46(1):93–94, 1993.

Dov Monderer and Lloyd Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996. URL https://EconPapers.repec.org/RePEc:eee:gamebe:v:14:y:1996:i:1:p:124-143.

John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950. ISSN 0027-8424. doi: 10.1073/pnas.36.1.48. URL https://www.pnas.org/content/36/1/48.

A. Nemirovski. On parallel complexity of nonsmooth convex optimization. *Journal of Complexity*, 10(4):451 – 463, 1994.

Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

Nicholas Pippenger. Sorting and selecting in rounds. *SIAM Journal on Computing*, 16(6):1032–1038, 1987.

Robert W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM J. Comput.*, 20(1):56–87, 1991. doi: 10.1137/0220004. URL https://doi.org/10.1137/0220004.

Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6 (1):1–114, 2012. doi: 10.2200/S00429ED1V01Y201207AIM018. URL https://doi.org/10.2200/S00429ED1V01Y201207AIM018.

Xiaoming Sun and Andrew Chi-Chih Yao. On the quantum query complexity of local search in two and three dimensions. *Algorithmica*, 55(3):576–600, 2009.

Craig Tovey. Polynomial local improvement algorithms in combinatorial optimization, 1981. Ph.D. thesis, Stanford University.

Leslie G. Valiant. Parallelism in comparison problems. *SIAM Journal on Computing*, 4(3):348–355, 1975.

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

Vijay V. Vazirani and Mihalis Yannakakis. Market equilibrium under separable, piecewise-linear, concave utilities. *J. ACM*, 58(3):10:1–10:25, 2011. doi: 10.1145/1970392.1970394. URL https://doi.org/10.1145/1970392.1970394.

Avi Wigderson and David Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. *Combinatorica*, 19(1):125–138, 1999.

Andrew Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

Shengyu Zhang. Tight bounds for randomized and quantum local search. *SIAM Journal on Computing*, 39(3):948–977, 2009.