Closing the Computational-Query Depth Gap in Parallel Stochastic Convex Optimization

Arun Jambulapati JMBLPATI@GMAIL.COM
University of Michigan

Aaron Sidford SIDFORD@STANFORD.EDU
Stanford University

Kevin Tian KJTIAN@CS.UTEXAS.EDU
University of Texas at Austin

Editors: Shipra Agrawal and Aaron Roth

Abstract

We develop a new parallel algorithm for minimizing Lipschitz, convex functions with a stochastic subgradient oracle. The total number of queries made and the query depth, i.e., the number of parallel rounds of queries, match the prior state-of-the-art, Carmon et al. (2023), while improving upon the computational depth by a polynomial factor for sufficiently small accuracy. When combined with previous state-of-the-art methods our result closes a gap between the best-known query depth and the best-known computational depth of parallel algorithms.

Our method starts with a *ball acceleration* framework of previous parallel methods, i.e., Carmon et al. (2020); Asi et al. (2021), which reduce the problem to minimizing a regularized Gaussian convolution of the function constrained to Euclidean balls. By developing and leveraging new stability properties of the Hessian of this induced function, we depart from prior parallel algorithms and reduce these ball-constrained optimization problems to stochastic unconstrained quadratic minimization problems. Although we are unable to prove concentration of the asymmetric matrices that we use to approximate this Hessian, we nevertheless develop an efficient parallel method for solving these quadratics. Interestingly, our algorithms can be improved using fast matrix multiplication and use nearly-linear work if the matrix multiplication exponent is 2.

The arXiv version of this paper can be found at https://arxiv.org/abs/2406.07373.

Keywords: Parallel optimization, stochastic convex optimization, acceleration, convolution

Acknowledgments

We thank Yair Carmon for helpful conversations during the initial stages of this project. Aaron Sidford was supported in part by a Microsoft Research Faculty Fellowship, NSF CAREER Award CCF-1844855, NSF Grant CCF1955039, and a PayPal research award. Part of this work was conducted while authors were visiting the Simons Institute for the Theory of Computing.

1. Introduction

Consider the classic problem of *Lipschitz convex optimization*. In this problem, there is a convex $f : \mathbb{R}^d \to \mathbb{R}$ that is 1-*Lipschitz*, i.e., $|f(x) - f(y)| \le \|x - y\|$ for all $x, y \in \mathbb{R}^d$, that is guaranteed to have a minimizer $x^\star \in \mathbb{R}^d$ with $\|x^\star\| \le 1$. The goal of the problem is to compute an (expected) ϵ-*approximate minimizer* to f, i.e., $x \in \mathbb{R}^d$ with $\mathbb{E} f(x) \le f(x^\star) + \epsilon$ given access to f only though a subgradient oracle g that when queried at $x \in \mathbb{R}^d$ outputs a vector $g(x) \in \partial f(x)$, where ∂f is the

set of subgradients of f at x. We focus on this standard setting in the introduction for simplicity, however our results extend to the more general case of bounded stochastic gradient oracles and further relaxations of the bounds on Lipschitz continuity and the minimizer (see Problem 1).

Lipschitz convex optimization is foundational in optimization theory, and its study has motivated well-known optimization algorithms. Simple, classic subgradient descent solves the problem with $O(\epsilon^{-2})$ oracle queries Nemirovski and Yudin (1983), and cutting plane methods solve the problem with $O(d \log \epsilon^{-1})$ oracle queries Khachiyan et al. (1988a). Consequently, the query complexity of the problem, i.e., the number of queries needed to solve the problem in the worst case, is $O(\min\{\epsilon^{-2}, d \log(\epsilon^{-1})\}$. Furthermore, this bound is known to be optimal among deterministic algorithms for all settings of ϵ and d Nemirovski and Yudin (1983), and is optimal even among randomized and quantum algorithms in certain settings Agarwal et al. (2012); Garg et al. (2021).

Due to the massive growth in dataset sizes and use of parallel computing resources, a line of work has studied *parallel* variants of Lipschitz convex optimization Nemirovski (1994); Duchi et al. (2012); Balkanski and Singer (2018); Bubeck et al. (2019); Carmon et al. (2023) and non-Euclidean generalizations Diakonikolas and Guzmán (2019); Chakrabarty et al. (2023). Study of this problem dates to at least Nemirovski (1994) which proposed the *parallel oracle access model*, in which the algorithm proceeds in T rounds and in round $t \in [T]$, the algorithm queries the oracle with n_t points $x_{t,1}, \ldots, x_{t,n_t} \in \mathbb{R}^d$ and receives the output of the oracle on each point. In round t, the n_t queried points can depend only on the queries in the previous rounds and the output of the oracle in those rounds (and additional randomness used by the algorithm). We call such an algorithm *highly parallel* Bubeck et al. (2019) if the number of queries in each round is bounded by a polynomial in d and a natural condition number for the problem, e.g., $n_t = \mathrm{poly}(d, \epsilon^{-1})$ for all $t \in [T]$. The total number of rounds of the algorithm, T, is called the *query depth* of the algorithm, and is a natural measure of its parallel performance.

Perhaps surprisingly, nontrivial parallel speedups, i.e., parallel algorithms whose query depth is better than the best-known query complexity, are only known for certain ϵ ranges. In fact, the $O(\epsilon^{-2})$ complexity of simple subgradient descent is optimal among highly parallel algorithms for sufficiently large ϵ. The associated lower bound was shown for all $\epsilon \gtrsim d^{-1/6}$ by Nemirovski (1994); Balkanski and Singer (2018) and for all $\epsilon \gtrsim d^{-1/4}$ by Bubeck et al. (2019).[1] Additionally, when $\epsilon \lesssim d^{-1}$, the current state-of-the-art query depth is achieved by applying classical cutting plane methods, e.g., Vaidya (1996). However, in the regime where $d^{-1/4} \gtrsim \epsilon \gtrsim d^{-1}$, which we term the *intermediate regime* of ϵ, nontrivial parallel speedups are known and there are algorithms which improve upon both subgradient descent and cutting plane methods. Namely, Lipschitz convex optimization was first shown to be solvable with query depth $\widetilde{O}(d^{1/4}\epsilon^{-1})$ by Duchi et al. (2012), and then $\widetilde{O}(d^{1/3}\epsilon^{-2/3})$ by Bubeck et al. (2019), the current state-of-the-art query depth.

However, beyond query depth, there are other natural ways to parameterize the complexity of a highly parallel algorithm. Specifically, we measure the complexity of parallel algorithms as follows.

Definition 1 (Parallel complexity) *We define the following four properties of an algorithm solving an optimization problem, e.g., Problem 1, with parallel access to an oracle g.*

1. Query depth*: number of sequential rounds of interaction with g (queries submitted in batch).*
2. Query complexity*: total number of queries to g.*

1. We use $\lesssim$, $\gtrsim$, and $\widetilde{O}$ to hide polylogarithmic factors in d and ϵ^{-1} in the introduction, and more broadly we use this notation to hide polylogarithmic factors in d and $\frac{LR}{\epsilon}$ throughout the paper in the context of Problem 1.

3. Computational depth*: number of sequential rounds of computation, outside of querying g.*

4. Computational complexity*: amount of computational work performed, outside of querying g.*

If g can be implemented with $O(\mathcal{T}_{\text{query}})$ work and $O(\mathcal{D}_{\text{query}})$ depth, we write that an algorithm can be implemented with $O(a \cdot \mathcal{D}_{\text{query}} + b)$ depth and $O(c \cdot \mathcal{T}_{\text{query}} + d)$ work when its query depth is $O(a)$, query complexity is $O(c)$, computational depth is $O(b)$, and computational complexity is $O(d)$.

Recently, Carmon et al. (2023) designed an algorithm which matched the $\widetilde{O}(d^{1/3}\epsilon^{-2/3})$ query depth of Bubeck et al. (2019), while simultaneously achieving a *query complexity* of $\widetilde{O}(d^{1/3}\epsilon^{-2/3} + \epsilon^{-2})$. This query complexity improved upon the $\widetilde{O}(d^{4/3}\epsilon^{-8/3})$ query complexity of Bubeck et al. (2019) and, when $\epsilon \lesssim d^{-1/4}$, matched that of subgradient descent, which is optimal for $\epsilon \gtrsim d^{-1/2}$ (as discussed earlier). Unfortunately, the *computational depth* of Carmon et al. (2023) is $\widetilde{O}(d^{1/4}\epsilon^{-1})$ (matching that of Duchi et al. (2012)). This computational depth scales polynomially worse than the query depth of Carmon et al. (2023) for $\epsilon \lesssim d^{-1/4}$, and is larger than the computational depth of state-of-the-art cutting plane methods, e.g., Vaidya (1996), when $\epsilon \lesssim d^{-3/4}$.

The key question motivating our work is whether this gap between the computational and query depths of state-of-the-art parallel algorithms in the intermediate regime is inherent. Specifically we address an open problem left by Carmon et al. (2023) as to whether there is an algorithm which, in the intermediate regime of ϵ, obtains the best-known query depth and query complexity, while simultaneously obtaining a computational depth no worse than its query depth (ideally at low overhead to the algorithm's computational complexity). Closing this gap is a natural problem that would expand the theory for parallel stochastic convex optimization, and potentially be of broader utility.

Our results. Our main result is a new algorithm which closes this gap for Lipschitz convex optimization and, more broadly, for stochastic convex optimization, as stated in Problem 1. The most general form of our result, Theorem 4, is stated in Section B. For simplicity in the introduction, we state the specialization of Theorem 4 to Lipschitz convex optimization here.

Theorem 2 *If queries to a subgradient oracle are implementable with $O(\mathcal{D}_{\text{query}})$ depth and $O(\mathcal{T}_{\text{query}})$ time, then there is a randomized algorithm which solves Lipschitz convex optimization with*

$$\widetilde{O}(d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}} \cdot \mathcal{D}_{\text{query}} + d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}}) \text{ depth and } \widetilde{O}\left((d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}} + \epsilon^{-2}) \cdot \mathcal{T}_{\text{query}} + d^{\frac{4}{3}}\epsilon^{-\frac{2}{3}} + d^{\frac{5-\omega}{3}}\epsilon^{-\frac{4\omega-2}{3}}\right) \text{ work,}$$

where $\omega < 2.372$ Alman et al. (2024) is such that multiplying $d \times d$ matrices requires $O(d^\omega)$ work.

Theorem 2 matches the query depth and query complexity of the state-of-the-art algorithm Carmon et al. (2023) (in terms of query depth) in the intermediate regime and attains a computational depth matching its query depth up to logarithmic factors (see Table 1 for a more complete comparison to prior work). Interestingly, the method uses fast matrix multiplication, a technique used to obtain state-of-the-art work and depth complexities for linear system solving; however, if $\omega = 2$, assuming vector operations using a stochastic gradient oracle require $\Omega(d)$ work, then its computational complexity is no worse than the work to query the oracle. Moreover, if $\mathcal{T}_{\text{query}}$ is moderately larger than d (e.g., $\mathcal{T}_{\text{query}} = \Omega(d \cdot \epsilon^{-\frac{1}{2}})$ for the current value of $\omega \neq 2$), the overhead of $d^{\frac{5-\omega}{3}}\epsilon^{-\frac{4\omega-2}{3}}$ is a low-order term compared to $\epsilon^{-2} \cdot \mathcal{T}_{\text{query}}$. In the more general Corollary 12 later in the paper, we

Method	Query depth	Query complexity	Computational depth
SGD Nemirovski and Yudin (1983)	ϵ^{-2}	ϵ^{-2}	ϵ^{-2}
Duchi et al. (2012)	$d^{\frac{1}{4}}\epsilon^{-1}$	$d^{\frac{1}{4}}\epsilon^{-1} + \epsilon^{-2}$	$d^{\frac{1}{4}}\epsilon^{-1}$
Bubeck et al. (2019)	$d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}}$	$d^{\frac{4}{3}}\epsilon^{-\frac{8}{3}}$	$d^{\frac{4}{3}}\epsilon^{-\frac{8}{3}}$
Carmon et al. (2023)	$d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}}$	$d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}} + \epsilon^{-2}$	$d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}} + d^{\frac{1}{4}}\epsilon^{-1}$
CPM* Vaidya (1996)	d	d	d
Theorem 2	$d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}}$	$d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}} + \epsilon^{-2}$	$d^{\frac{1}{3}}\epsilon^{-\frac{2}{3}}$

Table 1: **Highly parallel Lipschitz convex optimization algorithms.** The table depicts the history of improvements for solving Lipschitz convex optimization algorithms hiding polylogarithmic factors in d and ϵ^{-1}. CPM* refers to "cutting plane methods" and to the best of the authors knowledge Vaidya (1996) is the first paper to achieve the state-of-the-art complexity stated in the table; there are additional CPM results discussed in Theorem 3. The table applies to Problem 1 if each occurrence of ϵ^{-1} is replaced with $\kappa := \frac{LR}{\epsilon}$ and the CPM* line is changed (again, see Theorem 3).

show that it is possible obtain different tradeoffs between computational complexity and computational depth.

Beyond these quantitative improvements to parallel Lipschitz and stochastic convex optimization, to obtain our results, we provide several insights on related tools of potential independent interest (all outlined in Section 2). First, we provide a structural result (Lemma 9) about Gaussian convolutions of convex functions, a central tool in stochastic optimization. When combined with prior parallel optimization machinery, Lemma 9 reduces our problem to solving certain structured stochastic quadratic optimization problems in parallel. We then provide a new parallel algorithm for solving these quadratic optimization problems, which circumvents the need for concentration bounds. Along the way, we provide tools for boosting expected optimality bounds into high probability and handling hard constraints. We hope these tools may find broader use in optimization and learning theory and facilitate reaping the rewards of parallelism while mitigating the computational costs.

Remark 3 (Parallel complexity of cutting plane methods) *Cutting plane methods have a longer history than that conveyed in the CPM* line of Table 1. Levin (1965); Newman (1965) showed that it is possible to obtain query complexity $\widetilde{O}(d)$ and since then a line of work has established different tradeoffs between query complexity and computational complexity Shor (1977); Yudin and Nemirovskii (1976); Khachiyan (1980); Khachiyan et al. (1988b); Nesterov (1989); Vaidya (1996); Bertsimas and Vempala (2004); Lee et al. (2015); Jiang et al. (2020). Though computational depth was not necessarily highlighted in these works, we believe the first method with $\widetilde{O}(d)$ query complexity that could be implemented in depth $\widetilde{O}(d)$ is due to Vaidya (1996); subsequent papers may*

or may not have the same property. For stochastic convex optimization (Problem 1), though not explicitly stated, we believe the state-of-the-art is to leverage Vaidya (1996) within the framework of Sidford and Zhang (2023) to obtain an algorithm query and computational depth $\widetilde{O}(d)$ and query complexity $\widetilde{O}(d \cdot \mathrm{poly}(\kappa))$ for $\kappa := \frac{LR}{\epsilon}$.

Paper organization. We assemble the pieces to prove Theorem 2 and its generalization Theorem 4 throughout the rest of the paper. In Section 2, we overview our approach, by first reviewing facts about Gaussian convolutions and a ball acceleration result from Carmon et al. (2023), which constitutes our main framework. Additionally, we prove a result about the stability of Hessians of Gaussian convolutions of convex functions, which is the main structural insight enabling our new algorithms. We put together the pieces to prove our main result in Section 3.

Due to space constraints, we defer the formal statements and correctness proofs of various components of our algorithm to the appendices. In Appendix A, we first give an efficient parallel algorithm for optimizing (suitably regularized) local quadratic approximations to a Gaussian convolution. Finally, in Appendix B, we show how to obtain a key parallel ball optimization oracle implementation used in Theorem 4 by using this quadratic solver to implement the constrained ball oracles required by the acceleration framework by performing a binary search over our subroutine in Appendix A.

General notation. For $d \in \mathbb{N}$, we let $\mathbb{0}_d$ and $\mathbb{1}_d$ denote the all-zeroes and all-ones vectors in $\mathbb{R}^d$, $\mathbf{I}_d$ denote the identity matrix in $\mathbb{R}^{d \times d}$, and $[d] := \{i \in \mathbb{N} \mid 1 \le i \le d\}$. We let $\|\cdot\|$ denote the Euclidean norm of a vector. For $x \in \mathbb{R}^d$ we let $\mathbb{B}_x(r) := \{x' \in \mathbb{R}^d \mid \|x' - x\| \le r\}$ and $\mathbb{B}(r) := \mathbb{B}_{\mathbb{0}_d}(r)$ when d is clear from context. We use $\preceq$ to denote the Loewner partial ordering over $d \times d$ symmetric matrices, i.e., $\mathbf{A} \preceq \mathbf{B}$ if and only if $x^\top \mathbf{A} x \le x^\top \mathbf{B} x$ for all x, and define $\succeq$ analogously. For a positive semidefinite (PSD) $\mathbf{A} \in \mathbb{R}^{d \times d}$, we let $\|v\|_{\mathbf{A}} := (v^\top \mathbf{A} v)^{1/2}$ be the induced seminorm.

Parallel computation model. We assume a parallel computation model where all vector operations in $\mathbb{R}^d$ (e.g., addition and scalar multiplication) require $O(d)$ work and $O(1)$ depth, and that all matrix-vector multiplications (including computing dot products) require $O(\log d)$ depth. We let $\omega < 2.372$ Alman et al. (2024) be defined such that two $d \times d$ matrices can be multiplied with work $O(d^\omega)$. By a known reduction (Pan (1987); see also discussion in Pan and Reif (1985)), under this definition of ω, matrix multiplication can be performed in work $O(d^\omega)$ and depth $O(\log d)$. Under different parallel models, some of these bounds may incur polylogarithmic factor overheads.

2. Technical overview

In this remainder of the paper we consider the following stochastic convex optimization problem.

Problem 1 (Stochastic convex optimization) *In the* stochastic convex optimization *problem we are given $\epsilon, L, R > 0$ and access to a* stochastic gradient oracle $g : \mathbb{R}^d \to \mathbb{R}^d$ *satisfying, for all $x \in \mathbb{R}^d$, $\mathbb{E} g(x) \in \partial f(x)$ and $\mathbb{E}\|g(x)\|^2 \le L^2$ for convex $f : \mathbb{R}^d \to \mathbb{R}$. The goal is to output an* expected ϵ-approximate minimizer of f over $\mathbb{B}(R)$*, i.e., $x_{\mathrm{out}} \in \mathbb{R}^d$ such that $\mathbb{E} f(x_{\mathrm{out}}) \le \min_{x \in \mathbb{B}(R)} f(x) + \epsilon$. We assume g can be implemented with $O(\mathcal{T}_{\mathrm{query}})$ work and $O(\mathcal{D}_{\mathrm{query}})$ depth.*

Note that stochastic convex optimization (Problem 1) generalizes the Lipschitz convex optimization problem defined in Section 1. By Jensen's inequality and the convexity of $\|\cdot\|^2$, $\mathbb{E}\|g(x)\|^2 \le L^2$ implies that $\|\mathbb{E} g(x)\| \le L$ and consequently in Problem 1 at every point x there is a subgradient of

norm at most L. This implies that f is L-Lipschitz and thus Lipschitz convex optimization is the special case of Problem 1 when $L = R = 1$, each output of the stochastic subgradient oracle is deterministic, and f has a minimizer $x^\star \in \mathbb{R}^d$ with $\|x^\star\| \leq 1$.

Our main result is the following efficient parallel algorithm for solving Problem 1. This theorem immediately implies Theorem 2 in the special case of Lipschitz convex optimization.

Theorem 4 *There is an algorithm (*BallAccel *in Proposition 7, using Proposition 11 as a ball optimization oracle) which solves Problem 1 using:*

$$O\left(d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{13}{3}}(d\kappa)\log\log(d\kappa)\cdot\mathcal{D}_{\text{query}} + d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{28}{3}}(d\kappa)\right) \text{ depth, and}$$
$$O\left(\left(d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{10}{3}}(d\kappa) + \kappa^2\log^{\frac{19}{3}}(d\kappa)\right)\cdot\mathcal{T}_{\text{query}} + d^{\frac{4}{3}}\kappa^{\frac{2}{3}}\log^{\frac{10}{3}}(d\kappa) + d^{\frac{5-\omega}{3}}\kappa^{\frac{4\omega-2}{3}}\log^{\frac{19}{3}}(d\kappa)\right) \text{ work,}$$

where $\omega < 2.372$ Alman et al. (2024) is the matrix multiplication exponent, and $\kappa := \frac{LR}{\epsilon}$.

In the remainder of this technical overview we cover the main steps in proving Theorem 4, and discuss key insights and tools developed along the way, of possible broader utility. First, in Section 2.1 we provide an overview of the general framework used by both our parallel algorithm and prior work. In Section 2.2 we then discuss the key structural insight about this framework that we make and leverage to depart and improve upon prior work. In Section 2.3 we then discuss our main subroutine that we develop to leverage this structural insight and then in Section 2.4 we discuss implementing the subroutine in low computational depth to obtain our result.

2.1. Framework: convolutions and acceleration

All prior parallel improvements over subgradient descent for Problem 1 in the intermediate regime follow a similar broad framework Duchi et al. (2012); Bubeck et al. (2019); Carmon et al. (2023). Each of these works considers a process for smoothing f, i.e., working with a smooth approximation, and each uses accelerated optimization methods, i.e., some form of momentum, for optimizing the smoothing of f. Where the methods differ is in what smoothing is used, what accelerated method is applied, and how the accelerated method is implemented. Our method follows the approach of Carmon et al. (2023) which applies ball acceleration frameworks to optimize the Gaussian convolution of f. We begin by reviewing these techniques and highlighting their implication for parallel stochastic convex optimization.

Gaussian convolution. To solve Problem 1 rather than directly optimizing f, following the approach of Duchi et al. (2012); Bubeck et al. (2019); Carmon et al. (2023), we instead apply methods that optimize the Gaussian convolution of f, i.e., the function resulting from convolving f with a Gaussian, as defined below.

Definition 5 (Gaussian convolution) *For $f : \mathbb{R}^d \to \mathbb{R}$ and $\rho \geq 0$, we let f_ρ denote the convolution of f with $\mathcal{N}(\mathbb{0}_d, \rho^2\mathbf{I}_d)$, the normal distribution on $\mathbb{R}^d$ with covariance $\rho^2\mathbf{I}_d$ and mean $\mathbb{0}_d$. We use $*$ to denote convolution and γ_p to denote the density function of $\mathcal{N}(\mathbb{0}_d, \rho^2\mathbf{I}_d)$, so that $f_\rho = f * \gamma_\rho$ and*

$$f_\rho(x) := \mathbb{E}_{\xi\sim\mathcal{N}(\mathbb{0}_d,\rho^2\mathbf{I}_d)}\left[f(x-\xi)\right] = \int_{\mathbb{R}^d} f(x-\xi)\gamma_\rho(\xi)\mathrm{d}\xi \text{ for all } x \in \mathbb{R}^d.$$

Working with f_ρ offers a number of advantages: it is smooth, twice differentiable, and stochastic approximations to its gradient and Hessian can be computed by querying the stochastic gradient oracle at appropriately chosen random points. Additionally, it satisfies a new structural property we develop in Section 2.2. Formally, we recall the following facts from prior work Duchi et al. (2012); Bubeck et al. (2019), where by the Alexandrov theorem, the first and second derivatives are almost-everywhere defined in the third item. The fourth item in Fact 1 is shown in the proof of Lemma 8 in Bubeck et al. (2019).

Fact 1 (Lemma 8, Bubeck et al. (2019)) *For all convex, L-Lipschitz $f : \mathbb{R}^d \to \mathbb{R}$, $\rho \geq 0$, and $x \in \mathbb{R}^d$:*

1. *f_ρ is convex, L-Lipschitz, twice-differentiable, and satisfies $\nabla^2 f_\rho(x) \preceq \frac{L}{\rho}\mathbf{I}_d$,*
2. *$f(x) \leq f_\rho(x) \leq f(x) + L\rho\sqrt{d}$,*
3. *$\nabla f_\rho(x) = \int_{\mathbb{R}^d} \nabla f(x-\xi)\gamma_\rho(\xi)\mathrm{d}\xi$, and*
4. *$\nabla^2 f_\rho(x) = \int_{\mathbb{R}^d} \nabla^2 f(x-\xi)\gamma_\rho(\xi)\mathrm{d}\xi = \frac{1}{\rho^2}\int_{\mathbb{R}^d} \nabla f(x+\xi)\xi^\top \gamma_\rho(\xi)\mathrm{d}\xi$.*

In light of Fact 1, we make the following observation.

Observation 1 *In the context of Problem 1, let $\rho := \frac{\epsilon}{2L\sqrt{d}}$. If a point x_{out} solves an instance of Problem 1 with $f \leftarrow f_\rho$ and $\epsilon \leftarrow \frac{\epsilon}{2}$, then x_{out} also solves Problem 1 with $f \leftarrow f$ and $\epsilon \leftarrow \epsilon$.*

Proof By the first item in Fact 1, it is valid to let f be f_ρ in an instance of Problem 1. By the second item in Fact 1, letting $x^\star$ achieve $\min_{x\in\mathbb{B}(R)} f(x)$,

$$\mathbb{E}f(x_{\mathrm{out}}) \leq \mathbb{E}f_\rho(x_{\mathrm{out}}) \leq f_\rho(x^\star) + \frac{\epsilon}{2} \leq f(x^\star) + \epsilon.$$

■

In the rest of the paper, we consider a fixed instance of Problem 1; our approach is to optimize f by instead designing an algorithm for optimizing its Gaussian convolution f_ρ. In light of Observation 1, unless specified otherwise, we fix $\rho := \frac{\epsilon}{2L\sqrt{d}}$ throughout the rest of the paper.

Ball acceleration. To optimize the Gaussian convolution f_ρ, we leverage recent advances in accelerated proximal point algorithms, specifically a recent framework termed *ball acceleration* Carmon et al. (2020); Asi et al. (2021). Similar strategies were employed by Bubeck et al. (2019) and Carmon et al. (2023), which reduce solving Problem 1 to a smaller number of carefully-designed subproblems; in the case of Carmon et al. (2023), the subproblem is to minimize a regularized approximation to f_ρ over a small Euclidean ball (a *ball optimization oracle*). We specifically use the following variant of ball acceleration from Carmon et al. (2023).

Definition 6 (Ball optimization oracle) *We say $\mathcal{O}_{\mathrm{bo}}$ is a (ϕ, λ, r)*-ball optimization oracle *for $F : \mathbb{R}^d \to \mathbb{R}$ if given $\bar{x} \in \mathbb{R}^d$, $\mathcal{O}_{\mathrm{bo}}$ returns $x \in \mathbb{B}_{\bar{x}}(r)$ with*

$$\mathbb{E}\left[F(x) + \frac{\lambda}{2}\left\|x - \bar{x}\right\|^2\right] \leq \min_{x'\in\mathbb{B}_{\bar{x}}(r)}\left\{F(x) + \frac{\lambda}{2}\left\|x' - \bar{x}\right\|^2\right\} + \phi.$$

Proposition 7 (Proposition 2, Carmon et al. (2023)) *Let $F : \mathbb{R}^d \to \mathbb{R}$ be L-Lipschitz and convex, let $R > 0$, and let $x^\star \in \mathbb{B}(R)$. There is an algorithm* BallAccel *which takes parameters $r \in (0, R]$ and $\epsilon \in (0, LR]$ with the following guarantee. Define*

$$\kappa := \frac{LR}{\epsilon}, \; K := \left(\frac{R}{r}\right)^{\frac{2}{3}}, \quad \text{and } \lambda_\star := \frac{\epsilon K^2}{R^2} \log^2(K).$$

For a universal constant $C_{\text{ba}} > 0$, BallAccel *produces $x \in \mathbb{R}^d$ such that $\mathbb{E}F(x) \le F(x^\star) + \epsilon$. Letting $\mathcal{T}(\phi, \lambda, r) \ge d$ and $\mathcal{D}(\phi, \lambda, r) \ge \log(d)$ denote the work and depth used by a (ϕ, λ, r)-ball optimization oracle, the computational complexity of* BallAccel *is:*

$$C_{\text{ba}} K \log^3\left(\frac{R\kappa}{r}\right) \mathcal{T}\left(\frac{\lambda_\star r^2}{C_{\text{ba}}}, \frac{\lambda_\star}{C_{\text{ba}}}, r\right)$$
$$+ \sum_{j \in [\lceil \log_2 K + C_{\text{ba}} \rceil]} C_{\text{ba}} 2^{-j} K \log\left(\frac{R\kappa}{r}\right) \mathcal{T}\left(\frac{\lambda_\star r^2}{C_{\text{ba}} 2^j} \log^{-2}\left(\frac{R\kappa}{r}\right), \frac{\lambda_\star}{C_{\text{ba}}}, r\right),$$

and the depth of BallAccel *is:*

$$C_{\text{ba}} K \log^3\left(\frac{R\kappa}{r}\right) \mathcal{D}\left(\frac{\lambda_\star r^2}{C_{\text{ba}}}, \frac{\lambda_\star}{C_{\text{ba}}}, r\right)$$
$$+ \sum_{j \in [\lceil \log_2 K + C_{\text{ba}} \rceil]} C_{\text{ba}} 2^{-j} K \log\left(\frac{R\kappa}{r}\right) \mathcal{D}\left(\frac{\lambda_\star r^2}{C_{\text{ba}} 2^j} \log^{-2}\left(\frac{R\kappa}{r}\right), \frac{\lambda_\star}{C_{\text{ba}}}, r\right).$$

Implication and approach. Proposition 7 allows us to focus on designing ball optimization oracles for f_ρ in the remainder of the paper. Concretely, in our algorithm each oracle approximately solves a problem of the form

$$\min_{x \in \mathbb{B}_{\bar{x}}(r)} f_{\rho,\lambda,\bar{x}}(x) := f_\rho(x) + \frac{\lambda}{2} \|x - \bar{x}\|^2, \tag{1}$$

where f_ρ is the convolution of the density function of $\mathcal{N}(0, \rho^2 \mathbf{I}_d)$ and the function of interest f, and $\lambda > 0$ is a regularization parameter. The key challenge we address is how to implement a ball oracle for the Gaussian convolution efficiently in parallel. Since pioneering work of Duchi et al. (2012) it has been observed that stability of the Gaussian convolution can be useful in this endeavor, because higher moments of the Gaussian convolution can be efficiently approximated via parallel queries. Duchi et al. (2012) leveraged smoothness (stability of the gradient) in accelerated gradient descent, Bubeck et al. (2019) leveraged higher-order smoothness and concentration to approximate the Gaussian convolution over a large regions, and Carmon et al. (2023) leveraged how it is possible to obtain stochastic gradients for the Gaussian convolution at one point by sampling the stochastic gradient oracle a nearby point.

Our work exploits a new stability property of Gaussian convolutions (Lemma 9) that we introduce in the next Section 2.2. We leverage this property to essentially reduce implementing a ball oracle to solving a linear system induced by the Hessian of the Gaussian convolution, which we discuss how to do efficiently in parallel in Sections 2.3 and 2.4. Along the way, we introduce several tools which may be of broader interest to the stochastic optimization theory community.

2.2. Hessian stability of the Gaussian convolution

Our starting point for designing our ball optimization oracle is the observation that the Hessian of the convolved objective f_ρ is stable, in a precise sense, over balls of small radii $r \ll \rho$. To explain, note that for any $x, y \in \mathbb{R}^d$,

$$\nabla^2 f_\rho(x) = \int_{\mathbb{R}^d} \nabla^2 f(x-\xi)\gamma_\rho(\xi)\mathrm{d}\xi = \int_{\mathbb{R}^d} \nabla^2 f(y-\xi)\gamma_\rho(x-y+\xi)\mathrm{d}\xi.$$

So, if $\gamma_\rho(x-y+\xi) \approx \gamma_\rho(\xi)$ multiplicatively for all ξ, then similarly $\nabla^2 f_\rho(x) \approx \nabla^2 f_\rho(y)$. Unfortunately, this is not true: directly expanding shows

$$\frac{\gamma_\rho(x-y+\xi)}{\gamma_\rho(\xi)} = \exp\left(\frac{1}{2\rho^2}\left(\langle 2\xi, y-x\rangle - \|x-y\|^2\right)\right).$$

If ξ is large in the direction $y-x$, then the first term in the exponential dominates. Standard Gaussian tail bounds show the measure of such poorly-behaved ξ is small, but we do not have an a priori upper bound on $\nabla^2 f$ at the corresponding points (as f is possibly nonsmooth). Hence, it is unclear how to quantify the effect of these points. We leverage the simple observation that $f_\rho = f * \gamma_\rho$ is the convolution of $\gamma_{\rho/2}$ and that $f_{\rho/2}$ is a smooth function with a bounded Hessian. Consequently, $\nabla^2 f_\rho(x) \approx \nabla^2 f_\rho(y)$ does hold for x and y in a ball of small radii $r \ll \rho$ up to a small additive factor (which we can control by choosing the radii). To formalize this, we use the following notation for comparing deviations between a pair of PSD matrices in the following Theorem 8. We then bound the stability of Hessians of the Gaussian convolution in Theorem 9.

Definition 8 (Matrix approximation) *We say that PSD $\mathbf{A} \in \mathbb{R}^{d\times d}$ is an $(\epsilon_{\text{add}}, \epsilon_{\text{mul}})$-*approximation *to PSD $\mathbf{B} \in \mathbb{R}^{d\times d}$ if $\mathbf{A} \preceq \exp(\epsilon_{\text{mul}})\mathbf{B} + \epsilon_{\text{add}}\mathbf{I}_d$ and $\mathbf{B} \preceq \exp(\epsilon_{\text{mul}})\mathbf{A} + \epsilon_{\text{add}}\mathbf{I}_d$.*

We choose this Definition 8 because it is symmetric: $\mathbf{A}$ is an $(\epsilon_{\text{add}}, \epsilon_{\text{mul}})$-approximation of $\mathbf{B}$ if and only if $\mathbf{B}$ is an $(\epsilon_{\text{add}}, \epsilon_{\text{mul}})$-approximation of $\mathbf{A}$. This symmetry reflects our setting of comparing Hessians of the Gaussian convolution for pairs of points. It is straightforward that Definition 8 implies other notions of additive-multiplicative approximation, e.g., the less-symmetric alternative $\exp(-\epsilon_{\text{mul}})\mathbf{B} - \epsilon_{\text{add}}\mathbf{I}_d \preceq \mathbf{A} \preceq \exp(\epsilon_{\text{mul}})\mathbf{B} + \epsilon_{\text{add}}\mathbf{I}_d$.

Lemma 9 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and L-Lipschitz, and $\rho > 0$. Then for $x, y \in \mathbb{R}^d$, $\delta \in (0,1)$, $\nabla^2 f_\rho(x)$ is an $(\epsilon_{\text{add}}, \epsilon_{\text{mul}})$-approximation to $\nabla^2 f_\rho(y)$, following Definition 8, for*

$$\epsilon_{\text{mul}} := \frac{\|x-y\|^2}{\rho^2} + \frac{2\|x-y\|\sqrt{\log\frac{1}{\delta}}}{\rho},\ \epsilon_{\text{add}} := \frac{\sqrt{2}L\delta}{\rho}.$$

Proof Without loss of generality (as shifting by a constant vector does not affect the problem assumptions), let $y = \mathbb{0}_d$. Furthermore, let $g := f_\rho$ and $h := f_{\rho/\sqrt{2}}$, so that $g = h * \gamma_{\rho/\sqrt{2}}$. Note that by Fact 1, $\nabla^2 h$ is bounded by $\frac{\sqrt{2}L}{\rho}\mathbf{I}_d$ pointwise, and for all $\xi \in \mathcal{K}_r := \{\xi \in \mathbb{R}^d \mid \langle x, \xi\rangle \geq -r^2\}$,

$$\gamma_{\rho/\sqrt{2}}(\xi) = \exp\left(\frac{\|x\|^2 - 2\langle x, \xi\rangle}{\rho^2}\right)\gamma_{\rho/\sqrt{2}}(\xi - x) \leq \exp\left(\frac{\|x\|^2 + 2r^2}{\rho^2}\right)\gamma_{\rho/\sqrt{2}}(\xi - x). \quad (2)$$

We hence have, by Fact 1 applied to h and g,

$$\begin{aligned}\nabla^2 g(x) &= \int_{\mathcal{K}_r} \nabla^2 h(x-\xi)\gamma_{\rho/\sqrt{2}}(\xi)\mathrm{d}\xi + \int_{\mathbb{R}^d\setminus\mathcal{K}_r} \nabla^2 h(x-\xi)\gamma_{\rho/\sqrt{2}}(\xi)\mathrm{d}\xi \\ &\preceq \exp\left(\frac{\|x\|^2+2r^2}{\rho^2}\right)\int_{\mathcal{K}_r} \nabla^2 h(x-\xi)\gamma_{\rho/\sqrt{2}}(\xi-x)\mathrm{d}\xi \\ &+ \frac{\sqrt{2}L}{\rho}\left(\Pr_{\xi\sim\mathcal{N}(0,\frac{\rho^2}{2}\mathbf{I}_d)}[\xi\in\mathcal{K}_r]\right)\mathbf{I}_d \\ &\preceq \exp\left(\frac{\|x\|^2+2r^2}{\rho^2}\right)\nabla^2 g(\mathbb{0}_d) + \frac{\sqrt{2}L}{\rho}\exp\left(-\frac{r^4}{\rho^2\|x\|^2}\right)\mathbf{I}_d.\end{aligned}$$

In the second line, we used (2) and that $\nabla^2 h \preceq \frac{\sqrt{2}L}{\rho}\mathbf{I}_d$, and in the last line, we used standard tail bounds on the Gaussian error function (DLMF, Eq. 7.8.3). The conclusion follows by substituting the specific value $r = \sqrt{\rho\|x\|}\cdot\log^{\frac{1}{4}}(\frac{1}{\delta})$, and using symmetry of Theorem 8 in x and y. ■

Our ball optimization oracle objective $f_{\rho,\lambda,\bar{x}}$ in (1) is regularized, so when the additive term is dominated by the regularizer's Hessian, we can show $f_{\rho,\lambda,\bar{x}}$ is multiplicatively second-order stable. This is the key fact that we use to facilitate the implementation of our ball optimization oracles.

Corollary 10 *Let $\lambda > 0$ and suppose that $y \in \mathbb{B}_x(r)$ for $0 < r \le \frac{\rho}{6}\cdot\log^{-\frac{1}{2}}(\frac{2L}{\lambda\rho})$. Then, $\nabla^2 f_{\rho,\lambda,\bar{x}}(x)$ is a $(0,\log 2)$-approximation to $\nabla^2 f_{\rho,\lambda,\bar{x}}(y)$, following Definition 8.*

Proof Note that $x \in \mathbb{B}_y(r)$ if and only if $y \in \mathbb{B}_x(r)$, so by symmetry it suffices to show $\nabla^2 f_{\rho,\lambda,\bar{x}}(x) \preceq 2\nabla^2 f_{\rho,\lambda,\bar{x}}(y)$. Also, $\nabla^2 f_{\rho,\lambda,\bar{x}}(x) = \nabla^2 f_\rho(x) + \lambda\mathbf{I}_d$, and by Lemma 9 and our parameter settings,

$$\nabla^2 f_\rho(x) - \frac{\sqrt{2}L\delta}{\rho}\mathbf{I}_d \preceq 2\nabla^2 f_\rho(y) \text{ and } \lambda\mathbf{I}_d + \frac{\sqrt{2}L\delta}{\rho}\mathbf{I}_d \preceq 2\lambda\mathbf{I}_d,$$

for $\delta := \frac{\lambda\rho}{\sqrt{2}L}$. Adding the above two inequalities yields the conclusion. ■

2.3. Hessian optimization without Hessian approximation

Multiplicative Hessian stability in small balls was a key building block of algorithms in Carmon et al. (2020), which introduced ball acceleration. Specifically, Carmon et al. (2020) considered problems such as logistic and ℓ_p regression, where the objective's Hessian is both explicit and locally multiplicatively stable. This stability allows for efficient ball optimization via variants of Newton's method (e.g., gradient descent preconditioned by the objective's Hessian at the ball center).

In our setting, the use of such Newton's methods for ball optimization is complicated by two factors: our parallel implementation requirement, and the fact that we only have implicit access to $\nabla^2 f_{\rho,\lambda,\bar{x}}$, as it involves evaluating an integral. One useful characterization is that[2]

$$\nabla^2 f_\rho(\bar{x}) = \frac{1}{\rho^2}\int_{\mathbb{R}^d} \underbrace{\nabla f(x+\xi)\xi^\top}_{:=\mathbf{M}_\xi}\gamma_\rho(\xi)\mathrm{d}\xi,$$

2. The $\lambda\mathbf{I}_d$ component of $\nabla^2 f_{\rho,\lambda,\bar{x}}$ is explicit, so it suffices to evaluate $\nabla^2 f_\rho$.

so a natural way to proceed is to estimate $\nabla^2 f_\rho(\bar{x})$ as the average of a small number of randomly-sampled $\mathbf{M}_\xi$. Unfortunately, matrix concentration bounds such as the matrix Bernstein inequality (which are tight in the worst case Tropp (2015)) yield sample complexities which depend on

$$\max\left\{\underbrace{\left\|\mathbb{E}\left[\mathbf{M}_\xi\mathbf{M}_\xi^\top\right]\right\|_{\text{op}}}_{:=V_1}, \underbrace{\left\|\mathbb{E}\left[\mathbf{M}_\xi^\top\mathbf{M}_\xi\right]\right\|_{\text{op}}}_{:=V_2}\right\}, \tag{3}$$

for measuring convergence of random averages to $\nabla^2 f_\rho(\bar{x})$. In our case, V_2 can be significantly smaller than V_1: the former can be upper bounded by L^2 (as $\mathbb{E}\xi\xi^\top = \rho^2\mathbf{I}_d$), but the latter grows as $L^2 d$ (as $\mathbb{E}\xi^\top\xi = \rho^2 d$). A similar issue arises if one replaces the use of $\mathbf{M}_\xi$ with its symmetrized counterpart $\mathbf{S}_\xi := \frac{1}{2}(\mathbf{M}_\xi + \mathbf{M}_\xi^\top)$. This results in requiring at least d samples for Hessian approximation which is too many for our purposes (note that, e.g., the entire query complexity of Carmon et al. (2023) is $o(d)$ for moderate ϵ), and appears to be an obstacle for use of Newton's method.

We circumvent this obstacle by treating the implementation of each Newton step as a stochastic optimization problem, which breaks the symmetry between the dependence on V_1 and V_2. Specifically, each Newton iteration requires approximately solving a problem

$$\min_{x\in\mathbb{B}_{\bar{x}}(r)} \langle g, x\rangle + x^\top\nabla^2 f_\rho(\bar{x})x + \frac{\lambda}{2}\left\|x-\bar{x}\right\|^2, \tag{4}$$

for some vector g. We can therefore design a stochastic estimate $g + 2\mathbf{M}_\xi x$ of the gradient of the objective in (4), whose second moment only depends on $V_2 = \mathbb{E}\mathbf{M}_\xi^\top\mathbf{M}_\xi$ and not V_1. Interestingly, using $\mathbf{S}_\xi$ to estimate the gradient of (4) would run into the same issue as before (where the convergence rate also depends on $V_1 \approx L^2 d$), so using asymmetric estimates is crucial.

2.4. Parallel maintenance of rank-one updates

An additional challenge is implementing our strategy for solving (4) efficiently in parallel. We show in Section B.2 how to remove the constraint in (4) via a binary search procedure for an appropriate Lagrange multiplier, so it suffices to optimize over $\mathbb{R}^d$, subject to additional regularization. To facilitate this reduction, in Appendix A.3, we provide a technique for improving our expected error bounds to high probability bounds (similar to a recent technique in Sidford and Zhang (2023)).

With these reductions in place it suffices to solve unconstrained variants of (4) in parallel. In Appendix A.1, we provide a general stochastic composite gradient descent algorithm compatible with the stochastic oracles discussed in Section 2.3. It then turns out, as intended, that the resulting iterates of this stochastic composite gradient descent algorithm are highly-structured (as each $\mathbf{M}_\xi$ is rank-one). This structure is captured by the following linear algebraic maintenance problem, which we solve in Appendix A, allowing for the parallel implementation of our stochastic gradient method.

Problem 2 *Let $T \in \mathbb{N}$. For inputs $\{x_0, \{u_t, v_t, w_t\}_{t\in[T]}\} \subset \mathbb{R}^d$ and $\{c_t\}_{t\in[T]} \subset \mathbb{R}$, we wish to compute all $\{x_t\}_{t\in[T]}$ defined by the recurrence relation*

$$x_t := c_t\left(\left(\mathbf{I}_d - u_t v_t^\top\right)x_{t-1}\right) + w_t.$$

In our setting, c_t arises due to the regularization component, w_t captures the first-order part of (4), and u_t, v_t capture the (rank-one) estimate of the second-order part of (4). Note that if the term w_t did not exist, solving Problem 2 would amount to computing the product of T rank-one matrices in parallel, which can be done using a divide-and-conquer technique (see Lemma 16). By using a relatively lightweight combination of divide-and-conquer and fast matrix multiplication, we show that Problem 2 can similarly be solved in polylogarithmic depth and $O(dT^{\omega-1})$ work.

Applying this parallel implementation of our stochastic composite gradient descent algorithm, though our "in expectation-to-high probability" and binary search reductions, yields our ball optimization oracle implementation. When applied in the ball optimization framework to the Gaussian convolution, this then yields our main result, Theorem 4. While there are a few steps of indirection, we believe that reducing parallel optimization to stochastic quadratic optimization is an interesting and key contribution by itself. We hope the structural facts that enable this reduction and the algorithmic techniques that make it yield an efficient algorithm may have broader utility.

3. Proof of Theorem 4

We now give a short proof of Theorem 4, relying on the following parallel implementation of a ball optimization oracle proven in Appendix B, by combining the various ideas discussed in Section 2.

Proposition 11 *Define f_ρ as in Definition 5, where f is in the setting of Problem 1. Let $\lambda, r \in \mathbb{R}_{>0}$ satisfy $r \leq \frac{\rho}{6} \cdot \log^{-\frac{1}{2}}(\frac{2L}{\lambda\rho})$ and $\rho \leq \frac{L}{\lambda}$. For any $\phi \in (0, \frac{\lambda r^2}{100}]$, we can implement a (ϕ, λ, r)-ball optimization oracle (Definition 6) for f_ρ with*

$$O\left(\log\left(\frac{Lr}{\phi}\right)\log\log\left(\frac{Lr}{\phi}\right)\cdot\mathcal{D}_{\text{query}} + \frac{\lambda r^2}{\phi}\log^4\left(\frac{L^2}{\lambda\phi}\right)\log\left(\frac{dL^2}{\lambda\phi}\right)\right) \text{ depth,}$$

$$\text{and } O\left(\frac{L^2}{\lambda\phi}\log^5\left(\frac{L^2}{\lambda\phi}\right)\cdot\mathcal{T}_{\text{query}} + d\log^5\left(\frac{L^2}{\lambda\phi}\right)\cdot\frac{\lambda r^2}{\phi}\cdot\left(\frac{L^2}{\lambda^2 r^2}\right)^{\omega-1}\right) \text{ work.}$$

We can now prove our main result, Theorem 4, by combining the ball acceleration framework in Proposition 7 with our parallel ball optimization oracle in Proposition 11.

Theorem 4 *There is an algorithm (*BallAccel *in Proposition 7, using Proposition 11 as a ball optimization oracle) which solves Problem 1 using:*

$$O\left(d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{13}{3}}(d\kappa)\log\log(d\kappa)\cdot\mathcal{D}_{\text{query}} + d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{28}{3}}(d\kappa)\right) \text{ depth, and}$$

$$O\left(\left(d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{10}{3}}(d\kappa) + \kappa^2\log^{\frac{19}{3}}(d\kappa)\right)\cdot\mathcal{T}_{\text{query}} + d^{\frac{4}{3}}\kappa^{\frac{2}{3}}\log^{\frac{10}{3}}(d\kappa) + d^{\frac{5-\omega}{3}}\kappa^{\frac{4\omega-2}{3}}\log^{\frac{19}{3}}(d\kappa)\right) \text{ work,}$$

where $\omega < 2.372$ Alman et al. (2024) is the matrix multiplication exponent, and $\kappa := \frac{LR}{\epsilon}$.

Proof Throughout, let $\rho := \frac{\epsilon}{2L\sqrt{d}}$, and let $x^\star$ minimize f over $\mathbb{B}(R)$. We optimize f_ρ to expected error $\frac{\epsilon}{2}$, yielding the conclusion via Observation 1. Let $K = \Theta(d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{1}{3}}(d\kappa))$, and choose

$$\lambda_\star = \Theta\left(\frac{\epsilon\kappa^{\frac{4}{3}}d^{\frac{2}{3}}}{R^2}\log^2(d\kappa)\right),\ r = \Theta\left(\frac{\rho}{\sqrt{\log(d\kappa)}}\right), \tag{5}$$

to be compatible with the parameters in Proposition 7, such that $r \le \frac{\rho}{6} \cdot \log^{-\frac{1}{2}}(\frac{2C_{\text{ba}}L}{\lambda_\star \rho})$, following the notation in Proposition 7. This implies that Corollary 10 holds for every choice of $\lambda \ge \frac{\lambda_\star}{C_{\text{ba}}}$ used in ball optimization oracles by Proposition 7. Therefore, assuming $C_{\text{ba}} \ge 100$ without loss of generality, we can use Proposition 11 to implement every ball optimization oracle.

To bound the query depth, we apply Proposition 11 for each of the $O(K \log^3(d\kappa))$ ball optimization oracles required. To bound the query complexity, we have the claim from

$$\begin{aligned} O\left(K \log^3(d\kappa) \cdot \frac{L^2}{\lambda_\star^2 r^2} \log^5(d\kappa)\right) &= O\left(\kappa^2 \log^{\frac{16}{3}}(d\kappa)\right), \\ \sum_{j \in [\lceil \log_2 K + C_{\text{ba}} \rceil]} O\left(2^{-j} K \log(d\kappa) \cdot \frac{2^j L^2}{\lambda_\star^2 r^2} \log^7(d\kappa)\right) &= O\left(\frac{KL^2}{\lambda_\star^2 r^2} \log^9(d\kappa)\right) \\ &= O\left(\kappa^2 \log^{\frac{19}{3}}(d\kappa)\right). \end{aligned}$$

We also require one query per ball optimization oracle, so there is an additive $K \log^3(d\kappa)$ term. To bound the computational depth, we perform a similar calculation using Proposition 11:

$$\begin{aligned} O\left(K \log^3(d\kappa) \cdot \log^5(d\kappa)\right) &= O\left(d^{\frac{1}{3}} \kappa^{\frac{2}{3}} \log^{\frac{25}{3}}(d\kappa)\right), \\ \sum_{j \in [\lceil \log_2 K + C_{\text{ba}} \rceil]} O\left(2^{-j} K \log(d\kappa) \cdot 2^j \log^7(d\kappa)\right) &= O\left(K \log^9(d\kappa)\right) = O\left(d^{\frac{1}{3}} \kappa^{\frac{2}{3}} \log^{\frac{28}{3}}(d\kappa)\right). \end{aligned}$$

Finally, for the computational complexity, we have (using the bound $\omega \ge 2$)

$$\begin{aligned} O\left(K \log^3(d\kappa) \cdot d \log^5(d\kappa) \cdot \left(\frac{L^2}{\lambda_\star^2 r^2}\right)^{\omega-1}\right) &= O\left(d^{\frac{4}{3}} \kappa^{\frac{2}{3}} \log^{\frac{34}{3}-3\omega}(d\kappa) \cdot \left(\frac{\kappa^{\frac{4}{3}}}{d^{\frac{1}{3}}}\right)^{\omega-1}\right) \\ &= O\left(d^{\frac{5-\omega}{3}} \kappa^{\frac{4\omega-2}{3}} \log^{\frac{16}{3}}(d\kappa)\right), \end{aligned}$$

and

$$\sum_{j \in [\lceil \log_2 K + C_{\text{ba}} \rceil]} O\left(2^{-j} K \log(d\kappa) \cdot 2^j \cdot d \log^7(d\kappa) \left(\frac{L^2}{\lambda_\star^2 r^2}\right)^{\omega-1}\right) = O\left(d^{\frac{5-\omega}{3}} \kappa^{\frac{4\omega-2}{3}} \log^{\frac{19}{3}}(d\kappa)\right).$$

Again, we must perform at least one step per ball optimization oracle, so there is an additive $dK \log^3(d\kappa)$ term. Combining these bounds yields the conclusion. ■

Finally, we mention that the complexities in Proposition 11 admit a range of tradeoffs parameterized by a quantity $C \in [1, \frac{L^2}{\lambda^2 r^2}]$, which governs the number of consecutive iterations T we pass into our parallel solution to Problem 2, summarized in Corollary 32. By instead using a different parameter C as in Corollary 32, we obtain the following corollary, which interpolates between the two extremes of standard stochastic gradient descent and Theorem 4.

Corollary 12 *In the context of Theorem 4, for any $C \in [1, \frac{L^2}{\lambda_\star^2 r^2}]$ where $\lambda_\star, r$ are as defined in (5), there is an algorithm which solves Problem 1 using:*

$$O\left(d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{13}{3}}(d\kappa)\log\log(d\kappa)\cdot\mathcal{D}_{\text{query}} + Cd^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{28}{3}}(d\kappa)\right) \text{ depth,}$$
$$\text{and } O\left(\left(d^{\frac{1}{3}}\kappa^{\frac{2}{3}}\log^{\frac{10}{3}}(d\kappa) + \kappa^2\log^{\frac{19}{3}}(d\kappa)\right)\cdot\mathcal{T}_{\text{query}}\right)$$
$$+O\left(C^{2-\omega}\left(d^{\frac{4}{3}}\kappa^{\frac{2}{3}}\log^{\frac{10}{3}}(d\kappa) + d^{\frac{5-\omega}{3}}\kappa^{\frac{4\omega-2}{3}}\log^{\frac{19}{3}}(d\kappa)\right)\right) \text{ time.}$$

where $\omega < 2.372$ Williams et al. (2023) is the matrix multiplication exponent, and $\kappa := \frac{LR}{\epsilon}$.

References

Alekh Agarwal, Peter L. Bartlett, Pradeep Ravikumar, and Martin J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Trans. Inf. Theory*, 58(5):3235–3249, 2012.

Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. *CoRR*, abs/2404.16349, 2024.

Hilal Asi, Yair Carmon, Arun Jambulapati, Yujia Jin, and Aaron Sidford. Stochastic bias-reduced gradient methods. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021*, pages 10810–10822, 2021.

Eric Balkanski and Yaron Singer. Parallelization does not accelerate convex optimization: Adaptivity lower bounds for non-smooth convex minimization. *arXiv: 1808.03880*, 2018.

Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM (JACM)*, 51(4):540–556, 2004.

Sébastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, and Aaron Sidford. Complexity of highly parallel non-smooth convex optimization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, pages 13900–13909, 2019.

Yair Carmon and Oliver Hinder. The price of adaptivity in stochastic convex optimization. *CoRR*, abs/2402.10898, 2024.

Yair Carmon, Arun Jambulapati, Qijia Jiang, Yujia Jin, Yin Tat Lee, Aaron Sidford, and Kevin Tian. Acceleration with a ball optimization oracle. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, 2020.

Yair Carmon, Arun Jambulapati, Yujia Jin, Yin Tat Lee, Daogao Liu, Aaron Sidford, and Kevin Tian. Resqueing parallel and private stochastic convex optimization. *CoRR*, abs/2301.00457, 2023.

Deeparnab Chakrabarty, Andrei Graur, Haotian Jiang, and Aaron Sidford. Parallel submodular function minimization. *CoRR*, abs/2309.04643, 2023.

Jelena Diakonikolas and Cristóbal Guzmán. Lower bounds for parallel and randomized convex optimization. In *Conference on Learning Theory, COLT*, 2019.

DLMF. *NIST Digital Library of Mathematical Functions*. http://dlmf.nist.gov/, Release 1.1.8 of 2022-12-15. URL http://dlmf.nist.gov/. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds.

John C Duchi, Peter L Bartlett, and Martin J Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

Ankit Garg, Robin Kothari, Praneeth Netrapalli, and Suhail Sherif. No quantum speedup over gradient descent for non-smooth convex optimization. In *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 53:1–53:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

Arun Jambulapati, Victor Reis, and Kevin Tian. Linear-sized sparsifiers via near-linear time discrepancy theory. *CoRR*, abs/2305.08434, 2023.

Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 944–953, 2020.

Jonathan A. Kelner, Jerry Li, Allen X. Liu, Aaron Sidford, and Kevin Tian. Semi-random sparse recovery in nearly-linear time. In *The Thirty Sixth Annual Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 2352–2398. PMLR, 2023.

Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

Leonid G. Khachiyan, Sergei Pavlovich Tarasov, and I. I. Erlikh. The method of inscribed ellipsoids. *Soviet Math. Dokl.*, 37:226–230, 1988a.

Leonid G Khachiyan, Sergei Pavlovich Tarasov, and II Erlikh. The method of inscribed ellipsoids. In *Soviet Math. Dokl*, volume 37, pages 226–230, 1988b.

Simon Lacoste-Julien, Mark Schmidt, and Francis R. Bach. A simpler approach to obtaining an o(1/t) convergence rate for the projected stochastic subgradient method. *CoRR*, abs/1212.2002, 2012.

Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.

Anatoly Yur'evich Levin. An algorithm for minimizing convex functions. In *Doklady Akademii Nauk*, volume 160, pages 1244–1247. Russian Academy of Sciences, 1965.

A. Nemirovski and D.B̃. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, 1983.

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

Ju E Nesterov. Self-concordant functions and polynomial-time methods in convex programming. *Report, Central Economic and Mathematic Institute, USSR Acad. Sci*, 1989.

Donald J Newman. Location of the maximum on unimodal surfaces. *Journal of the ACM (JACM)*, 12(3):395–398, 1965.

Victor Y. Pan. Complexity of parallel matrix computations. *Theor. Comput. Sci.*, 54:65–85, 1987.

Victor Y. Pan and John H. Reif. Efficient parallel solution of linear systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 143–152. ACM, 1985.

Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics*, 13(1):94–96, 1977.

Aaron Sidford and Chenyi Zhang. Quantum speedups for stochastic optimization. *CoRR*, abs/2308.01582, 2023.

Joel A. Tropp. An introduction to matrix concentration inequalities. *Found. Trends Mach. Learn.*, 8(1-2):1–230, 2015.

Pravin M. Vaidya. A new algorithm for minimizing convex functions over convex sets. *Math. Program.*, 73:291–341, 1996. doi: 10.1007/BF02592216. URL https://doi.org/10.1007/BF02592216.

Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. *CoRR*, abs/2307.07970, 2023.

David B Yudin and Arkadi S Nemirovskii. Informational complexity and efficient methods for the solution of convex extremal problems. *Matekon*, 13(2):22–45, 1976.

Appendix A. Parallel optimization of quadratic subproblems

In this section, we develop an efficient parallel optimization method for solving structured unconstrained quadratics of the following form:

$$\min_{x\in\mathbb{R}^d} \langle g+v, x\rangle + \|x-z\|_{\mathbf{H}}^2 + \frac{\Lambda}{2}\|x\|^2, \text{ for } g,v\in\mathbb{R}^d,\ \mathbf{H}\in\mathbb{R}^{d\times d},\ \Lambda\in\mathbb{R}_{\geq 0}. \tag{6}$$

In particular, we consider a stochastic setting where instead of explicit access to g or $\mathbf{H}$ we assume sample access to random variables $\tilde{g}\in\mathbb{R}^d$ and $\widetilde{\mathbf{H}}\in\mathbb{R}^{d\times d}$ (not necessarily symmetric), such that

$$\mathbb{E}\tilde{g} = g \text{ and } \mathbb{E}\widetilde{\mathbf{H}} = \mathbf{H}. \tag{7}$$

Our consideration of this setting is motivated by the special case when

$$g = \nabla f_\rho(z) \text{ and } \mathbf{H} = \nabla^2 f_\rho(\mathbb{0}_d). \tag{8}$$

Objectives of the form (6), (8) arise in Newton's method for implementing a ball optimization oracle for $f_{\rho,\lambda,\bar{x}}$ as defined in (1), where $\bar{x}\leftarrow\mathbb{0}_d$ without loss of generality by shifting the problem domain. The additional quadratic $\langle v,x\rangle+\Lambda\|x\|^2$ arises due to regularization and a binary search on a Lagrange multiplier to enforce the domain constraint in (1). The two parts of this reduction (Newton's method and binary search) are respectively derived in Sections B.1 and B.2.

In Section A.1, we give an initial solver for the problem (6) that has an expected error guarantee and we show how to implement this solver in parallel in Section A.2. We then show how to boost this guarantee to hold with high probability using a reduction we develop in Section A.3. Finally, we assemble these components to give the main exports of this section in Section A.4.

A.1. Composite stochastic optimization

In this section, we fix $\Lambda\in\mathbb{R}_{\geq 0}$ and $v,z\in\mathbb{R}^d$ throughout, and decompose (6) into two parts:

$$h(x) := h_1(x)+h_2(x) \text{ where } h_1(x) := \langle g,x\rangle + \|x-z\|_{\mathbf{H}}^2 \text{ and } h_2(x) := \langle v,x\rangle + \frac{\Lambda}{2}\|x\|^2\ . \tag{9}$$

We treat the objective in (6) as a composite objective h_1+h_2, where we can exactly optimize over h_2, and we have stochastic access to h_1. Specifically, we use that $g_1(x) := \tilde{g}+2\widetilde{\mathbf{H}}(x-z)$ is an unbiased estimate of $\nabla h_1(x)$. More broadly, we design an algorithm for minimizing h_1+h_2 under the assumption that for some $L_1,L_2\in\mathbb{R}_{\geq 0}$,

$$\mathbb{E}\left[\|g_1(x)\|^2\right] \leq L_1^2 + L_2^2\|x-z\|^2 \text{ for all } x\in\mathbb{R}^d. \tag{10}$$

To motivate (10), Fact 1 shows that in the setting of Problem 1, when $g,\mathbf{H}$ are as in (8), we can use

$$g_1(x) = g(z-\xi) + \frac{2}{\rho^2}\langle \xi, x-z\rangle\, g(\xi) \text{ for } \xi\sim\mathcal{N}(\mathbb{0}_d,\rho^2\mathbf{I}_d) \tag{11}$$

as our unbiased estimator. We give a second moment bound on the estimator in (11).

Lemma 13 *In the setting of Problem 1, for $x,z\in\mathbb{R}^d$, where $\mathbb{E}$ is taken over $\xi\sim\mathcal{N}(\mathbb{0}_d,\rho^2\mathbf{I}_d)$ and the randomness of querying g at $z-\xi$ and ξ,*

$$\mathbb{E}\left[\left\|g(z-\xi)+\frac{2}{\rho^2}\langle\xi,x-z\rangle\, g(\xi)\right\|^2\right] \leq 2L^2 + \frac{8L^2}{\rho^2}\|x-z\|^2.$$

Proof Let $a := g(z-\xi)$ and $b := \frac{2}{\rho^2}\langle \xi, x-z\rangle\, g(\xi)$, and note that $\mathbb{E}\,\|a+b\|^2 \le 2\mathbb{E}\,\|a\|^2 + 2\mathbb{E}\,\|b\|^2$. Further, by definition $\mathbb{E}\,\|a\|^2 \le L^2$, so it suffices to bound $\mathbb{E}\,\|b\|^2$. We conclude by computing:

$$\begin{aligned}\mathbb{E}\left[\|\langle \xi, x-z\rangle\, g(\xi)\|^2\right] &= \mathbb{E}_{\xi\sim\mathcal{N}(\mathbb{0}_d,\rho^2\mathbf{I}_d)}\left[\mathbb{E}\left[\|g(\xi)\|^2 \mid \xi\right]\langle \xi, x-z\rangle^2\right] \\ &\le L^2\mathbb{E}_{\xi\sim\mathcal{N}(\mathbb{0}_d,\rho^2\mathbf{I}_d)}\left[\langle \xi, x-z\rangle^2\right] = L^2\rho^2\,\|x-z\|^2\,.\end{aligned}$$

■

In other words, in the setting of Problem 1, the assumption (10) holds with $L_1^2 = 2L^2$ and $L_2^2 = \frac{8L^2}{\rho^2}$. We now move to the abstraction of (9), (10), and design a general-purpose algorithm for this optimization problem. At the end of the section, we specialize our method to Problem 1.

Due to the unconstrained nature of our problem and the dependence (10) on movement from z, we take care to ensure that the iterates of our algorithm do not drift by too much. This is the primary challenge faced in this setting. We give an algorithm (Algorithm 1) and analysis based on Lacoste-Julien et al. (2012), using a "warm-started" step size schedule to ensure sufficient expected norm bounds on iterates.

Algorithm 1: UnconstrainedSGD(h_2, z, g_1)

Input: $h_2 : \mathbb{R}^d \to \mathbb{R}$, a Λ-strongly convex function, $z \in \mathbb{R}^d$, and $g_1 : \mathbb{R}^d \to \mathbb{R}^d$, an unbiased estimator for ∇h_1 where $h_1 : \mathbb{R}^d \to \mathbb{R}$ is convex, and for all $x \in \mathbb{R}^d$, (10) holds.

$x_0 \leftarrow \operatorname{argmin}_{x\in\mathbb{R}^d} h_2(x)$

$T_0 \leftarrow \frac{8L_2^2}{\Lambda^2}$

for $0 \le t < T$ **do**

  $\eta_t \leftarrow \frac{2}{\Lambda(t+T_0)}$

  $x_{t+1} \leftarrow \arg\min_{x\in\mathbb{R}^d}\{\eta_t\langle g_1(x_t), x\rangle + \eta_t h_2(x) + \frac{1}{2}\|x - x_t\|^2\}$

end

Return: $x_{\text{avg}} := \frac{1}{T}\sum_{0\le t<T} x_t$

Lemma 14 *Let $h_1 : \mathbb{R}^d \to \mathbb{R}$, $h_2 : \mathbb{R}^d \to \mathbb{R}$, $g_1 : \mathbb{R}^d \to \mathbb{R}^d$, and $z \in \mathbb{R}^d$ satisfy the assumptions of Algorithm 1 for $L_2 \ge \Lambda$, and let $x^\star$ minimize $h := h_1 + h_2$. Following notation of Algorithm 1,*

$$\mathbb{E}\left[h(x_{\text{avg}})\right] - h(x^\star) \le \left(\frac{2L_2^2}{\Lambda T} + \frac{\Lambda}{4T}\right)\|x_0 - x^\star\|^2 + \frac{\log(T+T_0)}{\Lambda T}\left(L_1^2 + 2L_2^2\,\|z-x^\star\|^2\right).$$

Proof Throughout the proof, for all $0 \le t < T$ let

$$D_t := \mathbb{E}\left[\frac{1}{2}\|x_t - x^\star\|^2\right] \text{ and } \Phi_t := \mathbb{E}\left[h(x_t)\right] - h(x^\star).$$

By first-order optimality of x_{t+1},

$$\begin{aligned}&\langle g_1(x_t), x_{t+1} - x^\star\rangle + \langle \nabla h_2(x_{t+1}), x_{t+1} - x^\star\rangle \\ &\le \frac{1}{2\eta_t}\left(\|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2 - \|x_t - x_{t+1}\|^2\right).\end{aligned}$$

By rearranging and taking an expectation (conditioned on the realization of x_t),

$$\begin{aligned} h_1(x_t) + h_2(x_{t+1}) - h(x^\star) + \frac{\Lambda}{2}\left\|x_{t+1} - x^\star\right\|^2 &\le \langle \nabla h_1(x_t), x_t - x^\star\rangle + \langle \nabla h_2(x_{t+1}), x_{t+1} - x^\star\rangle \\ &\le \frac{1}{2\eta_t}\left(\left\|x_t - x^\star\right\|^2 - \mathbb{E}\left\|x_{t+1} - x^\star\right\|^2\right) \\ &+ \mathbb{E}\left[\langle g_1(x_t), x_t - x_{t+1}\rangle + \frac{1}{2\eta_t}\left\|x_t - x_{t+1}\right\|^2\right] \\ &\le \frac{1}{2\eta_t}\left(\left\|x_t - x^\star\right\|^2 - \mathbb{E}\left\|x_{t+1} - x^\star\right\|^2\right) \\ &+ \frac{\eta_t}{2}\mathbb{E}\left[\left\|g_1(x_t)\right\|^2\right]. \end{aligned}$$

In the first inequality, we used convexity of h_1 and strong convexity of h_2, and in the last line, we applied the Cauchy-Schwarz and Young's inequalities to bound the quantity in the third line. Hence, iterating expectations, rearranging, and using the assumption (10),

$$\begin{aligned} \Phi_t + \Lambda D_{t+1} &\le \mathbb{E}\left[h_2(x_t) - h_2(x_{t+1})\right] + \frac{1}{\eta_t}\left(D_t - D_{t+1}\right) + \frac{\eta_t}{2}\left(L_1^2 + L_2^2\mathbb{E}\left[\left\|x_t - z\right\|^2\right]\right) \\ &\le \mathbb{E}\left[h_2(x_t) - h_2(x_{t+1})\right] + \frac{1}{\eta_t}\left(D_t - D_{t+1}\right) + \frac{\eta_t}{2}\left(L_1^2 + 2L_2^2\left\|z - x^\star\right\|^2 + 4L_2^2 D_t\right). \end{aligned}$$

Moreover, note that for the given choice of parameters, i.e., using $T_0 = \frac{8L_2^2}{\Lambda^2}$,

$$\begin{aligned} \frac{1}{\eta_t} + 2\eta_t L_2^2 &\le \frac{\Lambda(t+T_0)}{2} + \frac{4L_2^2}{\Lambda T_0} = \frac{\Lambda(t+T_0+1)}{2}, \\ \frac{1}{\eta_t} + \Lambda &= \frac{\Lambda(t+T_0)}{2} + \Lambda = \frac{\Lambda(t+T_0+2)}{2}. \end{aligned}$$

Combining the above two displays, we have

$$\begin{aligned} \Phi_t &\le \mathbb{E}\left[h_2(x_t) - h_2(x_{t+1})\right] + \frac{\Lambda(t+T_0+1)}{2}D_t - \frac{\Lambda(t+T_0+2)}{2}D_{t+1} \\ &+ \frac{\eta_t}{2}\left(L_1^2 + 2L_2^2\left\|z - x^\star\right\|^2\right). \end{aligned}$$

Therefore, by telescoping over T iterations, and using minimality of x_0 with respect to h_2,

$$\frac{1}{T}\sum_{0\le t<T}\Phi_t \le \frac{\Lambda(T_0+1)}{4T}\left\|x_0 - x^\star\right\|^2 + \left(\frac{1}{T}\sum_{0\le t<T}\frac{\eta_t}{2}\right)\left(L_1^2 + 2L_2^2\left\|z - x^\star\right\|^2\right).$$

Applying convexity of h and plugging in our parameter choices yields the claim, where we bound the partial harmonic sequence $\sum_{0\le t<T}\frac{1}{t+T_0} \le \sum_{t=8}^{T+T_0}\frac{1}{t} \le \log(t+T_0)$. ■

We now state Algorithm 2, our specialization of Algorithm 1 to the problem (6), (8). In Line 2 of Algorithm 2, we used the definition of h_2 from (9). We observe that the update in Line 2 can be conveniently written in closed form as

$$x_{t+1} \leftarrow \frac{1}{1+\eta_t\Lambda}\left(x_t - \eta_t v - \eta_t g_t' - \frac{2\eta_t}{\rho^2}\left\langle \xi_t, x_t - z\right\rangle g_t\right). \tag{12}$$

We demonstrate in Section A.2 how to support efficient parallel maintenance of weighted averages of iterates undergoing updates of the form (12). For now, we give an error bound following Lemma 14.

Algorithm 2: UnconstrainedSGDConv$(\Lambda, \rho, z, v, T, f)$

Input: $\Lambda, \rho \geq 0$, $z, v \in \mathbb{R}^d$, $T \in \mathbb{N}$, f in the setting of Problem 1
$x_0 \leftarrow -\frac{1}{2\Lambda} v$
$T_0 \leftarrow \frac{64L^2}{\rho^2\Lambda^2}$
for $0 \leq t < T$ **do**
  $\xi_t \sim \mathcal{N}(\mathbb{0}_d, \rho^2 \mathbf{I}_d)$
  $g_t \leftarrow g(\xi_t)$, $g_t' \leftarrow g(z - \xi_t)$
end
for $0 \leq t < T$ **do**
  $\eta_t \leftarrow \frac{2}{\Lambda(t+T_0)}$
  $x_{t+1} \leftarrow \arg\min_{x \in \mathbb{R}^d} \{\eta_t \langle g_t' + \frac{2}{\rho^2} \langle \xi_t, x_t - z \rangle g_t, x \rangle + \eta_t h_2(x) + \frac{1}{2} \|x - x_t\|^2\}$
end
Return: $x_{\text{avg}} := \frac{1}{T} \sum_{0 \leq t < T} x_t$

Corollary 15 *Following the notation in (9) and Algorithm 2, let $x^\star_{\Lambda,z,v}$ minimize (6) under the setting (8), and suppose $\max(\|z\|, \|x_0\|) \leq \rho$ and $\rho \leq \frac{L}{\Lambda}$. Then,*

$$\mathbb{E} h(x_{\text{avg}}) - h(x^\star_{\Lambda,z,v}) \leq \left(\frac{66L^2 \log(T + T_0)}{\Lambda T} + \frac{\Lambda \rho^2}{2T} \right) \left(1 + \frac{\left\| x^\star_{\Lambda,z,v} \right\|^2}{\rho^2} \right).$$

Proof In light of Lemma 13 and the fact that the gradient estimator in (11) is unbiased for ∇h_1 defined in (9), we can apply Lemma 14 with h_1, h_2 as in (9), and $L_1^2 := 2L^2$ and $L_2 := 8L^2/\rho^2$. The conclusion follows from Lemma 13 once we simplify using $\max(\|x_0\|, \|z\|) \leq \rho$. ■

To gain some intuition for Corollary 15, note that it shows if $\|x^\star_{\Lambda,z,v}\| \ll \rho$ and our target error is $\Theta(\Lambda r^2)$, we recover the standard $\frac{L^2}{\Lambda T}$ rate of strongly convex optimization under a bounded-variance gradient oracle, up to a log factor. We show how to combine this guarantee with a binary search in Section B to efficiently solve constrained optimization problems, as required by Proposition 7.

A.2. Parallel maintenance of rank-1 updates

In this section, we give our parallel solution to Problem 2, reproduced here for convenience.

Problem 2 *Let $T \in \mathbb{N}$. For inputs $\{x_0, \{u_t, v_t, w_t\}_{t \in [T]}\} \subset \mathbb{R}^d$ and $\{c_t\}_{t \in [T]} \subset \mathbb{R}$, we wish to compute all $\{x_t\}_{t \in [T]}$ defined by the recurrence relation*

$$x_t := c_t \left(\left(\mathbf{I}_d - u_t v_t^\top \right) x_{t-1} \right) + w_t.$$

Our updates in Algorithm 2, as stated in (12), are exactly of the form in Problem 2, with

$$\begin{aligned} c_t \leftarrow \frac{1}{1+\eta_{t-1}\Lambda},\ u_t \leftarrow \frac{2\eta_{t-1}}{\rho^2} g_{t-1},\ v_t \leftarrow \xi_{t-1}, \\ w_t \leftarrow -c_t \left(\eta_{t-1}(v + g'_{t-1}) + \frac{2\eta_{t-1}}{\rho^2} \langle \xi_{t-1}, z \rangle \, g_{t-1} \right). \end{aligned} \tag{13}$$

Moreover, all of the $\{c_t, u_t, v_t, w_t\}_{t\in[T]}$ can be queried and precomputed with $\widetilde{O}(1)$ depth and $O(dT)$ work. Accordingly, it suffices to solve Problem 2 to give a parallel implementation. As a warmup to our overall solution, we first give our parallel solution to the following simpler Problem 3.

Problem 3 *In the setting of Problem 2, we wish to compute x_T.*

We then modify our strategy to solve Problem 2, at a slightly larger parallel depth. Our solution to Problem 3 follows straightforwardly from maintaining matrix products in a dyadic fashion, using the following observation (Lemma 16) on maintaining low-rank updates of the identity. We combine this with a variant of a parallel prefix sum maintenance strategy for recursive matrix-vector products.

Lemma 16 *Let $\mathbf{A}_0, \mathbf{B}_0, \mathbf{A}_1, \mathbf{B}_1 \in \mathbb{R}^{d\times r}$ for $r \in [d]$. In depth $O(\log d)$ and work $O(dr^{\omega-1})$, we can compute $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d\times 2r}$ such that $\mathbf{I}_d - \mathbf{A}\mathbf{B}^\top = (\mathbf{I}_d - \mathbf{A}_0\mathbf{B}_0^\top)(\mathbf{I}_d - \mathbf{A}_1\mathbf{B}_1^\top)$.*

Proof Note that it suffices to choose

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{A}_1 - \mathbf{A}_0\mathbf{B}_0^\top\mathbf{A}_1 \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} \mathbf{B}_0 & \mathbf{B}_1 \end{pmatrix}.$$

The bottleneck is evaluating $\mathbf{A}_0\mathbf{B}_0^\top\mathbf{A}_1$ which takes work $O(dr^{\omega-1})$ and depth $O(\log d)$. ■

Building upon Lemma 16, we next give a solution to Problem 3 when all $c_t = 1$.

Lemma 17 *If $c_t = 1$ for all $t \in [T]$, we can solve Problem 3 with depth $O(\log(d)\log(T))$ and work $O(dT^{\omega-1})$.*

Proof Throughout this proof let $\ell := \lfloor \log_2 T \rfloor + 1$, define $w_0 := x_0$, and for $t > T$ let $u_t = v_t = w_t := \mathbb{0}_d$. We also define for $s \le t$, where all matrix products are evaluated right-to-left,

$$\mathbf{M}_{t:s} := \prod_{r=s}^{t} \left(\mathbf{I}_d - u_r v_r^\top \right),$$

so that $\mathbf{M}_{T:1} = (\mathbf{I}_d - u_T v_T^\top) \ldots (\mathbf{I}_d - u_1 v_1^\top)$. As all iterates after T do not change, we observe that

$$x_T = x_{2^\ell} = \sum_{t=0}^{2^\ell - 1} \mathbf{M}_{2^\ell:t+1} w_t, \tag{14}$$

since by definition, $w_{2^\ell} = \mathbb{0}_d$. For notational convenience, we define

$$\mathbf{M}_{(i,j)} := \mathbf{M}_{2^i j:2^i(j-1)+1} \tag{15}$$

for each $0 \le i \le \ell$ and $j \in [2^{\ell-i}]$. We observe that with $O(dT^{\omega-1})$ work, we can explicitly compute $\{\mathbf{A}_{(i,j)}, \mathbf{B}_{(i,j)}\}_{0\le i\le \ell, j\in[2^{\ell-i}]}$ such that

$$\mathbf{M}_{(i,j)} = \mathbf{I}_d - \mathbf{A}_{(i,j)}\mathbf{B}_{(i,j)}^\top, \text{ for } \mathbf{A}_{(i,j)}, \mathbf{B}_{(i,j)} \in \mathbb{R}^{d\times 2^i}.$$

To see this, we clearly can compute all $\{\mathbf{A}_{(0,j)}, \mathbf{B}_{(0,j)}\}_{j\in[2^\ell]}$ with $O(dT)$ work and $O(1)$ depth. Further, for $0 \le i < \ell$, assuming access to all $\{\mathbf{A}_{(i,j)}, \mathbf{B}_{(i,j)}\}_{j\in[2^{\ell-i}]}$, we can apply Lemma 16 in parallel to compute each required $\mathbf{A}_{(i+1,j)}, \mathbf{B}_{(i+1,j)}$ with $O(d(2^i)^{\omega-1})$ work, incurring a total work of

$$2^{\ell-i} \cdot O\left(d(2^i)^{\omega-1}\right) = O(dT) \cdot (2^i)^{\omega-2}.$$

Summing over all $0 \le i \le \ell$ yields a geometric sequence with dominant term $O(dT^{\omega-1})$ as desired. This procedure can be implemented in depth $O(\log(T)\log(d))$ by repeatedly applying Lemma 16. Next, for each $0 \le i \le \ell$ and $j \in [2^{\ell-i}]$, define

$$x_{(i,j)} := \sum_{k\in[2^i]} \mathbf{M}_{2^ij:2^i(j-1)+k} w_{2^i(j-1)-1+k}, \tag{16}$$

such that by inspection, the following recursion holds for $i \in [\ell]$:

$$x_{(i,j)} = \mathbf{M}_{2^ij:2^ij-2^{i-1}+1} x_{(i-1,2j-1)} + x_{(i-1,2j)}. \tag{17}$$

For example,

$$\begin{aligned} x_{(3,1)} &= \mathbf{M}_{8:1}w_0 + \mathbf{M}_{8:2}w_1 + \mathbf{M}_{8:3}w_2 + \mathbf{M}_{8:4}w_3 + \mathbf{M}_{8:5}w_4 + \mathbf{M}_{8:6}w_5 + \mathbf{M}_{8:7}w_6 + \mathbf{M}_{8:8}w_7 \\ &= \mathbf{M}_{8:5}\underbrace{\left(\mathbf{M}_{4:1}w_0 + \mathbf{M}_{4:2}w_1 + \mathbf{M}_{4:3}w_2 + \mathbf{M}_{4:4}w_3\right)}_{x_{(2,1)}} \\ &+ \underbrace{\mathbf{M}_{8:5}w_4 + \mathbf{M}_{8:6}w_5 + \mathbf{M}_{8:7}w_6 + \mathbf{M}_{8:8}w_7}_{x_{(2,2)}}. \end{aligned}$$

Our goal is simply to compute $x_{(\ell,1)} = x_T$, where we recall (14). First, we clearly can compute all $x_{(0,j)}$ for $j \in [2^\ell]$ with $O(dT)$ work. Further, for $0 \le i < \ell$, assuming access to all $x_{(i,j)}$ for $j \in [2^{\ell-i}]$ and all $\{\mathbf{A}_{(i,j)}, \mathbf{B}_{(i,j)}\}_{j\in[2^{\ell-i}]}$, we claim we can compute all $x_{(i+1,j)}$ for $j \in [2^{\ell+1-i}]$ in parallel incurring a total work of $O(dT)$. To see this, to compute each $x_{(i+1,j)}$ via the recursion (17), we require one vector addition, and one matrix-vector multiplication through

$$\mathbf{M}_{2^ij:2^ij-2^{i-1}+1} = \mathbf{I}_d - \mathbf{A}_{(i-1,2j)}\mathbf{B}_{(i-1,2j)}^\top$$

which can be performed in time $O(d2^i)$. Therefore, the total work required to compute all $x_{(i+1,j)}$ is bounded by $O(d2^\ell) = O(dT)$. Finally, summing over all $0 \le i \le \ell$ the total work of these computations is bounded by $O(dT\log T)$ which does not asymptotically dominate the work. The depth of this recursive computation is again bounded by $O(\log(T)\log(d))$. ■

We conclude with a simple extension of Lemma 17 to general $\{c_t\}_{t\in[T]}$, giving our full solution.

Corollary 18 *We can solve Problem 3 with depth $O(\log(T)\log(d))$ and work $O(dT^{\omega-1})$.*

Proof First, we may assume all $\{c_t\}_{t\in[T]}$ are nonzero, else we can begin the recursion in Problem 3 starting from the index right after the last zero value. Under this assumption, by writing $C_t := \prod_{s\in[t]} c_s$ and $x_t = C_t y_t$ for all $t \geq 0$, we have the equivalent recurrence

$$y_t = C_t^{-1}\left(c_t\left(\mathbf{I}_d - u_t v_t^\top\right)C_{t-1}y_{t-1} + w_t\right) = \left(\mathbf{I}_d - u_t v_t^\top\right)y_{t-1} + C_t^{-1}w_t.$$

Further, computing all $\{C_t\}_{t\in[T]}$ in the same dyadic fashion used to compute the $\mathbf{M}_{(i,j)}$ in Lemma 17 can be performed in $O(\log T)$ depth and $O(T\log T)$ work. Hence it suffices to apply Lemma 17 to an instance of Problem 3 with all $c_t = 1$ and inputs

$$\{x_0, \{u_t, v_t, C_t^{-1}w_t\}_{t\in[T]}\}$$

and multiply the final output vector (corresponding to y_T) by C_T. The scalings $\{C_t^{-1}w_t\}_{t\in[T]}$ can be performed using $O(1)$ depth and $O(dT)$ work by computing each in parallel. ■

We now show how modifying the strategy for Problem 3 also yields an efficient parallel solution for the generalization in Problem 2. As before, we first handle the case where all $c_t = 1$.

Lemma 19 *If $c_t = 1$ for all $t \in [T]$, we can solve Problem 2 with depth $O(\log^2(T)\log(d))$ and work $O(dT^{\omega-1})$.*

Proof We follow notation of Lemma 17, and assume that in depth $O(\log(T)\log(d))$ and work $O(dT^{\omega-1})$, we have precomputed all $\mathbf{M}_{(i,j)}$ and $x_{(i,j)}$ for $0 \leq i \leq \ell$ and $j \in [2^{\ell-i}]$. Define $z_s := x_s - w_s$ for all $s \in [2^\ell]$, and let $\mathcal{T}(\ell)$ be the total work it takes to compute all $\{z_s\}_{s\in[2^\ell]}$ in an instance of Problem 2, given access to all $\mathbf{M}_{(i,j)}$ and $x_{(i,j)}$ defined in (15) and (16). In particular, (14) holds with the left-hand side replaced with z_s and the right-hand side's summation ending at $s-1$. We claim

$$\mathcal{T}(\ell) = 2\mathcal{T}(\ell-1) + O(dT\log T) \implies \mathcal{T}(\ell) = O(dT\log^2 T), \tag{18}$$

which gives the total work bound because adding w_s to each z_s can be done in constant depth and $O(dT)$ work which does not dominate. Clearly, computing all $\{z_s\}_{s\in[2^{\ell-1}]}$ can be done within work $\mathcal{T}(\ell-1)$. Moreover, for each $2^{\ell-1} < s \leq 2^\ell$, note that

$$z_s = \underbrace{\sum_{t=0}^{2^{\ell-1}-1}\mathbf{M}_{s:t+1}w_t}_{:=u_s} + \underbrace{\sum_{t=2^{\ell-1}}^{s-1}\mathbf{M}_{s:t+1}w_t}_{:=v_s}.$$

Computing all $\{v_s\}_{2^{\ell-1}<s\leq 2^\ell}$ can be done within work $\mathcal{T}(\ell-1)$, as these constitute an independent copy of the problem over $2^{\ell-1}$ iterations. Finally, we complete the proof of (18) by showing we can compute all $\{u_s\}_{2^{\ell-1}<s\leq 2^\ell}$ using $O(dT\log T)$ work and $O(\log^2(T)\log(d))$ depth. Define $u^\star := \sum_{t=0}^{2^{\ell-1}-1}\mathbf{M}_{2^{\ell-1}:t+1}w_t$, and note that

$$u_s = \mathbf{M}_{s:2^{\ell-1}+1}u^\star \text{ for all } 2^{\ell-1} < s \leq 2^\ell.$$

Since we have access to all the $\mathbf{M}_{(i,j)}$, we can compute the u_s in a dyadic fashion, i.e., we first compute $u_{2^{\ell-1}+2^{\ell-2}}$ and u_{2^ℓ} using a single matrix multiplication each, and then $u_{2^{\ell-1}+2^{\ell-3}}$ and

$u_{2^{\ell-1}+3\cdot 2^{\ell-3}}$, and so on. The work cost of multiplying by a matrix $\mathbf{M}_{(i,j)}$ is $O(d2^i)$, so the overall work of computing all $\{u_s\}_{2^{\ell-1}<s\le 2^\ell}$ is then indeed bounded by

$$O(d2^{\ell-1}) + O(2 \cdot d2^{\ell-2}) + O(2^2 \cdot d2^{\ell-3}) + \ldots = O(dT \log T),$$

as claimed. To see the depth bound, we can solve the two instances of $\mathcal{T}(\ell-1)$ independently, leading to a recursion depth of $O(\log T)$. The sequential depth of each recursion layer (due to computing the $\{u_s\}_{2^{\ell-1}<s\le 2^\ell}$) is bottlenecked by $O(\log(T)\log(d))$ due to the use of $O(\log T)$ matrix multiplications, each of which takes depth $O(\log d)$. Thus, overall the depth is $O(\log^2(T)\log(d))$. ■

By using the same reduction as in Corollary 18, we extend our solution in Lemma 19 to the general setting of Problem 2, giving our main result.

Proposition 20 *We can solve Problem 2 with depth $O(\log^2(T)\log(d))$ and work $O(dT^{\omega-1})$.*

Proof We first consider the case where all $\{c_t\}_{t\in[T]}$ are nonzero. Define the sequence $\{y_t\}_{t\in[T]}$ as in Corollary 18, which can be computed within the depth and work budgets given by Lemma 19. Since

$$x_t = C_t y_t \text{ for all } t \in [T],$$

it suffices to compute all $\{C_t\}_{t\in[T]}$ and perform the scalings $C_t y_t$ in parallel, which can be done within the stated budgets by the arguments of Corollary 18. Finally, in the case where some $c_t = 0$, we can split the problem into independent contiguous blocks of nonzero c_t values whose total sizes sum to at most T and which can be solved in parallel. Since the claimed work is superlinear in T, it remains correct after operating on each contiguous block separately. ■

As a consequence of Proposition 20, we have the following complexity bounds on Algorithm 2.

Corollary 21 *Following the notation in (9) and Algorithm 2, let $S \in [T]$ be arbitrary. We can implement T iterations of Algorithm 2 using*

$$O\left(\mathcal{D}_{\text{query}} + \frac{T}{S} \cdot \log^2(S)\log(d)\right) \text{ depth, and } O\left(T \cdot \mathcal{T}_{\text{query}} + dTS^{\omega-2}\right) \text{ work.}$$

Proof Assume for simplicity that S divides T, else we can obtain the result by increasing T by a constant factor. Recall from (12), (13) that implementing Algorithm 2 is an instance of Problem 2, where we are required to compute the average iterate. Moreover we can compute all the inputs to Problem 2 in parallel, which gives the query depth and query complexity. Our strategy is to use Proposition 20 for every S consecutive iterations, which gives us all the iterates in the stated computational depth and complexity by applying Proposition 20, $\frac{T}{S}$ times. Given all the iterates we can clearly output the average within the stated computational depth and complexity. ■

A.3. High-probability error bound reduction

We now give a simple reduction from an algorithm which returns an approximate minimizer with high probability, to one which returns an expected approximate minimizer. Our reduction assumes access to a bounded-variance gradient estimator. We note that a similar procedure appears as Section 4.2 of Sidford and Zhang (2023), but it does not quite suffice for our purposes due to the composite nature of our objective. We provide a different proof for completeness, which also shows a slightly stronger fact that the approximately-optimal point returned comes from the original set of candidates.

Finally, we note that our reduction has implications on the query complexity of high-probability stochastic convex optimization (i.e., Problem 1) in the non-parallel setting. In particular, it shows that the expected error guarantee in Problem 1 can be boosted to have failure probability $\leq \delta$ at a $\mathrm{polylog}(\frac{1}{\delta})$ overhead in the query complexity. Such a result is classical when $g(x)$ satisfies stronger tail bounds (such as a sub-Gaussian norm), but to our knowledge the corresponding result in the bounded variance setting (as in Problem 1) was only obtained recently by Carmon and Hinder (2024). We do note that Carmon and Hinder (2024)'s approach yields an improved polylogarithmic overhead in $\frac{1}{\delta}$, which they show is optimal; we find it interesting to explore if different tradeoffs in Proposition 22 yield the same optimal result.

Proposition 22 *Let $h : \mathbb{R}^d \to \mathbb{R}$ be differentiable with minimizer $x^\star$, and assume $h(x) = h_1(x) + h_2(x)$ for all $x \in \mathbb{R}^d$ and we can evaluate $h_2(x)$ for any $x \in \mathbb{R}^d$. Further, suppose $S := \{x_i\}_{i\in[k]}$ has $\|x_i - x_j\| \leq R$ for all $i, j \in [k]$, and $\min_{i\in[k]} h(x_i) - h(x^\star) \leq \epsilon$. Finally, suppose $g_1(x)$ is an unbiased estimate for $\nabla h_1(x)$ and $\mathbb{E}[\|g_1(x)\|^2] \leq L^2$ for all x in the convex hull of S. For $\delta \in (0, 1)$, there is an algorithm which returns $x \in S$ with $h(x) - h(x^\star) \leq 2\epsilon$ with probability $\geq 1 - \delta$, using*

$$O\left(\frac{L^2R^2}{\epsilon^2} \cdot k \log\left(\frac{\log k}{\delta}\right)\right) \text{ queries to } g_1 \text{ and } k \text{ evaluations of } h_2.$$

The query depth used is $O(\log(k))$, and the computational depth used is $O((\log(d)+\log\log(\frac{k}{\delta}))\log(k))$.

Proof Fix any two $i, j \in [k]$ with $i \neq j$. Our first step is to build a high-probability approximation subroutine for the value of $h_1(x_i) - h_1(x_j)$. To this end, observe that for $x_{i,j}^{(t)} := (1-t)x_i + tx_j$,

$$h_1(x_j) - h_1(x_i) = \underbrace{\int_0^1 \left\langle \nabla h_1(x_{i,j}^{(t)}), x_j - x_i \right\rangle \mathrm{d}t}_{:=I(i,j)} = \mathbb{E}_{t\sim_{\text{unif.}}[0,1]}\left[\left\langle \nabla h_1(x_{i,j}^{(t)}), x_j - x_i \right\rangle\right].$$

Next, consider the estimator

$$Z(i,j) := \left\langle g_1(x_{i,j}^{(t)}), x_j - x_i \right\rangle, \text{ for } t \sim_{\text{unif.}} [0,1].$$

From the given assumptions, it is clear that $\mathbb{E}Z(i,j) = I(i,j)$ and

$$\mathbb{E}Z(i,j)^2 \leq \mathbb{E}\left[\left\|g_1(x_{i,j}^{(t)})\right\|^2 \|x_j - x_i\|^2\right] \leq L^2R^2.$$

Therefore, Chebyshev's inequality shows that averaging $\frac{4L^2R^2}{\Delta^2}$ independent copies of $Z(i,j)$ produces a Δ-additive approximation of $I(i,j)$ with probability $\frac{3}{4}$. A median of $O(\log(\frac{\log k}{\delta}))$ such independent averages then estimates $I(i,j)$ to additive error Δ with probability at least $1-\frac{\delta}{\log k}$, by a Chernoff bound. The total computation required to produce this estimate for a pair $(i,j) \in [k]\times[k]$ is

$$O\left(\frac{L^2R^2}{\Delta^2}\cdot\log\left(\frac{\log k}{\delta}\right)\right) \text{ calls to } g_1.$$

Using two additional evaluations of h_2, we can thus estimate $h(x_j)-h(x_i)$ to additive error Δ, with probability $\geq 1-\frac{\delta}{\log k}$. To obtain the claim, we run a tournament on the elements of S using our subroutine as an approximate comparator. Suppose that k is a power of 2 without loss of generality (otherwise we can duplicate x_1 appropriately), and initialize a complete binary tree on $2k-1$ nodes (with depth $\log_2(k)$), placing elements of S at the leaf nodes. We define the i^{th} layer of the tree to be all nodes which are distance exactly i from the leaf nodes (the leaf nodes themselves form layer 0). Starting from layer 1 and working upwards, for a node in layer ℓ with children x_i and x_j, we compute $E(i,j)$, a $\Delta_\ell = \frac{\epsilon}{3}(\frac{4}{3})^{-\ell}$-approximation to $h(x_i)-h(x_j)$, and promote the child with smaller estimated h value (i.e., we promote x_i if $E(i,j)\leq 0$, and we promote x_j otherwise). Assume without loss of generality that x_1 minimizes $h(x)$ over S. Conditioned on all estimates on x_1's root-to-leaf path succeeding (which happens with probability $1-\delta$ since there are $\log k$ nodes), the minimum function value on level ℓ is at most $h(x_1)+\sum_{i\in[\ell]}\frac{\epsilon}{3}(\frac{4}{3})^{-i}$, and so the algorithm returns some node y with $h(y)\leq \min_{i\in[k]}h(x_i)+\epsilon \leq h(x^\star)+2\epsilon$. The complexity and correctness follow by setting $\Delta \leftarrow \frac{\epsilon}{\log_2(k)}$. The total failure probability follows from a union bound, since there are at most $k-1$ comparisons (as each comparison eliminates an element).

We now control the number of gradients computed by the algorithm. Level ℓ of the tree calls the estimation subroutine $k2^{-\ell}$ times with failure probability $\frac{\delta}{\log k}$ and error $\frac{\epsilon}{3}(\frac{4}{3})^{-\ell}$: summing over all layers gives a total gradient bound of

$$\begin{aligned} O(1)\sum_{\ell\in[\log_2 k]}\frac{L^2R^2k2^{-\ell}}{\epsilon^2(\frac{4}{3})^{-2\ell}}\log\left(\frac{\log k}{\delta}\right) &= O\left(\frac{L^2R^2}{\epsilon^2}k\log\left(\frac{\log k}{\delta}\right)\right)\sum_{\ell\in[\log_2 k]}\left(\frac{8}{9}\right)^\ell \\ &= O\left(\frac{L^2R^2}{\epsilon^2}k\log\left(\frac{\log k}{\delta}\right)\right). \end{aligned}$$

■

A.4. Putting it all together

In this section, we combine the tools given in the previous sections to develop two high-probability optimization primitives, which will be used in Section B.2 in conjunction with a binary search to give our overall ball optimization oracle implementation. We now formally define the two types of oracles we require for implementing our binary search. Roughly speaking, the first type of oracle (Definition 23) is used to find a range of regularization amounts α such that the resulting regularized minimizers lie in a ball of radius $O(r)$. The second type of oracle (Definition 24) is then used to obtain accurate minima for our original constrained function. In the following definitions, for a fixed differentiable convex function F and $\alpha > 0$, we let

$$x^\star_\alpha := \text{argmin}_{x\in\mathbb{R}^d} F(x) + \frac{\alpha}{2}\|x\|^2. \tag{19}$$

Definition 23 *We call $\mathcal{O}_1$ an (r, β)-*phase-one oracle *for $F : \mathbb{R}^d \to \mathbb{R}$ if on input $\alpha \geq \beta$, following notation (19), $\mathcal{O}_1$ returns x satisfying*

$$\|x - x_\alpha^\star\| \leq \frac{r + \|x_\alpha^\star\|}{100}.$$

Definition 24 *We call $\mathcal{O}_2$ a (Δ, r, β)-*phase-two oracle *for $F : \mathbb{R}^d \to \mathbb{R}$ if on input $\alpha \geq \beta$, following notation (19), $\mathcal{O}_2$ returns x satisfying*

$$F(x) + \frac{\alpha}{2}\|x\|^2 \leq F(x_\alpha^\star) + \frac{\alpha}{2}\|x_\alpha^\star\|^2 + \Delta.$$

We specialize the following discussion to the specific context where F is of the form

$$\begin{aligned} &\text{argmin}_{x \in \mathbb{B}(r)} \langle \nabla f_{\rho,\lambda}(z), x \rangle + \|x - z\|^2_{\nabla^2 f_\rho(\mathbb{0}_d)} + \lambda \|x - z\|^2 \\ &= \text{argmin}_{x \in \mathbb{B}(r)} \langle \nabla f_\rho(z) - \lambda z, x \rangle + \|x - z\|^2_{\nabla^2 f_\rho(\mathbb{0}_d)} + \lambda \|x\|^2, \end{aligned} \tag{20}$$

where $\|z\| \leq r$ and $\rho \geq r$, $\lambda > 0$. These constrained subproblems arise in an approximate Newton's method which we develop in Section B.1. Formally, we define

$$F(x) := \langle \nabla f_\rho(z) - \lambda z, x \rangle + \|x - z\|^2_{\nabla^2 f_\rho(\mathbb{0}_d)}. \tag{21}$$

We use two tools to boost constant-accuracy bounds to high probability. The first is the reduction in Proposition 22, and the second is the following standard geometric aggregation method.

Lemma 25 (Claim 1, Kelner et al. (2023)) *Let $\delta \in (0, 1)$ and $x \in \mathbb{R}^d$ be unknown, and let $\mathcal{A}$ be an algorithm which returns $x' \in \mathbb{R}^d$ such that $\|x' - x\| \leq \frac{\Delta}{3}$ with probability $\geq \frac{2}{3}$ in $\mathcal{D}_\mathcal{A}$ depth and $\mathcal{T}_\mathcal{A}$ work. There is an algorithm which returns y such that $\|y - x\| \leq \Delta$ with probability $\geq 1 - \delta$, using $O(\mathcal{D}_\mathcal{A} + \log(d))$ depth and $O(\mathcal{T}_\mathcal{A} \cdot \log(\frac{1}{\delta}) + d\log^2(\frac{1}{\delta}))$ work.*

We now state our oracle implementations and their guarantees.

Lemma 26 *Let F be defined as in (21), assume $\rho \leq \frac{L}{\lambda}$, and let $\delta \in (0, 1)$. We can implement an $(r, 2\lambda)$-phase-one oracle for F which succeeds with probability $\geq 1 - \delta$ with*

$$O\left(\mathcal{D}_{\text{query}} + \log^2\left(\frac{L}{\lambda r}\right)\log(d)\right) \text{ depth,}$$

and $O\left(\left(\frac{L^2 \log\left(\frac{L}{\lambda r}\right)}{\lambda^2 r^2}\right)\log\left(\frac{1}{\delta}\right) \cdot \mathcal{T}_{\text{query}} + d\left(\frac{L^2 \log\left(\frac{L}{\lambda r}\right)}{\lambda^2 r^2}\right)^{\omega - 1}\log\left(\frac{1}{\delta}\right) + d\log^2\left(\frac{1}{\delta}\right)\right)$ *work.*

Proof We first show how to obtain x such that $\|x - x_\alpha^\star\| \leq \frac{r + \|x_\alpha^\star\|}{300}$ with probability $\geq \frac{2}{3}$. By Markov's inequality and α-strong convexity of $F(x) + \frac{\alpha}{2}\|x\|^2$, it is enough to produce x such that

$$\mathbb{E}\left[\left(F(x) + \frac{\alpha}{2}\|x\|^2\right) - \left(F(x_\alpha^\star) + \frac{\alpha}{2}\|x_\alpha^\star\|\right)^2\right] \leq \frac{\alpha}{6} \cdot \frac{(r + \|x_\alpha^\star\|^2)}{300^2}.$$

In the context of Corollary 15 (with $\Lambda \leftarrow \alpha$), it suffices to take

$$T = O\left(\frac{\rho^2}{r^2} + \frac{L^2 \log\left(\frac{L}{\alpha r}\right)}{\alpha^2 r^2}\right) = O\left(\frac{L^2 \log\left(\frac{L}{\alpha r}\right)}{\alpha^2 r^2}\right).$$

The conclusion follows from Corollary 21 (with $S \leftarrow T$) and Lemma 25. ■

Lemma 27 *Let F be defined as in (21), assume $\rho \in [r, \frac{L}{\lambda}]$, and let $\delta \in (0,1)$. For $\Delta \le \frac{\lambda r^2}{100}$, we can implement a $(\Delta, r, \max(\alpha_{3r}, 2\lambda))$-phase-two oracle for f which succeeds with probability $\ge 1-\delta$ with*

$$O\left(\log\log\left(\frac{1}{\delta}\right)\cdot\mathcal{D}_{\text{query}} + \frac{\lambda r^2}{\Delta}\log\left(\frac{L^2}{\lambda\Delta}\right)\log^2\left(\frac{L}{\lambda r}\right)\log\left(\frac{d}{\delta}\right)\right) \text{ depth,}$$

and $$O\left(\frac{L^2\log\left(\frac{L^2}{\lambda\Delta}\right)}{\lambda\Delta}\log^3\left(\frac{1}{\delta}\right)\cdot\mathcal{T}_{\text{query}} + d\log^4\left(\frac{L^2}{\delta\lambda\Delta}\right)\cdot\frac{\lambda r^2}{\Delta}\cdot\left(\frac{L^2}{\lambda^2 r^2}\right)^{\omega-1}\right) \text{ work.}$$

Proof Since $\alpha \ge \alpha_{3r}$, we can produce a point x with expected suboptimality $\frac{\Delta}{6}$ to the function $F(x) + \frac{\alpha}{2}\|x\|^2$ by calling Corollary 15 with

$$T = O\left(\frac{\alpha\rho^2}{\Delta} + \frac{L^2\log\left(\frac{L^2}{\alpha\Delta}\right)}{\alpha\Delta}\right) = O\left(\frac{L^2\log\left(\frac{L^2}{\alpha\Delta}\right)}{\alpha\Delta}\right).$$

Therefore, by Markov's inequality x has suboptimality $\frac{\Delta}{2}$ with probability $\ge \frac{2}{3}$. Moreover, each x which achieves this suboptimality has, by α-strong convexity,

$$\|x - x_\alpha^\star\| \le \sqrt{\frac{\Delta}{\alpha}} \le \frac{r}{10}.$$

We run this algorithm $k = O(\log\frac{1}{\delta})$ times, where the constant is large enough that Lemma 25 applies with probability $\ge 1 - \frac{\delta}{3}$, and also $k \ge \log_3(\frac{3}{\delta})$, so some run produces x with suboptimality gap $\frac{\Delta}{2}$ with probability $\ge 1 - \frac{\delta}{3}$. Call $A = \{x_i\}_{i\in[k]}$ the set of output points, and let $x_{i^\star} \in S$ satisfy

$$F(x_{i^\star}) + \frac{\alpha}{2}\|x_{i^\star}\|^2 - F(x_\alpha^\star) - \frac{\alpha}{2}\|x_\alpha^\star\|^2 \le \frac{\Delta}{2}.$$

By Lemma 25, we obtain $\bar{x}$ with $\|\bar{x} - x_\alpha^\star\| \le 3\sqrt{\Delta/\alpha}$. Let $B \subseteq A$ be the elements of A with $\|\bar{x} - x\| \le 4\sqrt{\Delta/\alpha}$, which contains $x_{i^\star}$ by definition. Then for any $x, x' \in B$, we have

$$\|x - x'\| \le 8\sqrt{\frac{\Delta}{\alpha}}.$$

Moreover, since all points in B lie at distance $\le \frac{2r}{5}$ from $x_\alpha^\star$, their norms are all at most $4r$. Since $\|z\| \le r$ by assumption, Lemma 13 shows we can implement an unbiased estimator for the gradient of the implicit part of (21), i.e., $\nabla f_\rho(z) + 2\nabla^2 f_\rho(\mathbb{0}_d)(x - z)$, with second moment $O(L^2)$, for any x in the convex hull of B. We therefore can apply Proposition 22 with $\epsilon \leftarrow \frac{\Delta}{2}$ to obtain an element of B with suboptimality gap $\le \Delta$ with probability $\ge 1 - \frac{\delta}{3}$, within the stated complexities. We can check that all other steps also fall within the stated complexities, using Corollary 21 with $S \leftarrow \frac{L^2}{\alpha\lambda r^2}$. ■

Appendix B. Parallel stochastic convex optimization

In this section, we prove Theorem 2 by using the results of Section A to implement the ball optimization oracles required by Proposition 7. Our reduction from (constrained) ball optimization to the (unconstrained) quadratic problems considered by Section A proceeds in two steps.

1. In Section B.1, we show how to use Hessian stability of the ball optimization oracle subproblems (Corollary 10) to efficiently solve these problems using an approximate Newton's method.

2. The subproblems required by our method in Section B.1 are constrained optimization problems, which are almost compatible with our tools in Section A. In Section B.2, we develop a simple binary search procedure, inspired by a procedure in Jambulapati et al. (2023), which reduces each constrained optimization problem to a small number of unconstrained stochastic optimization problems.

B.1. Approximate Newton's method

In this section, we state and analyze an approximate variant of a constrained Newton's method under Hessian stability, patterned off classical analyses of gradient descent in a quadratic norm.

Algorithm 3: ConstrainedNewton$(\lambda, T, x_0, f, \phi)$

Input: Positive definite $\mathbf{A} \in \mathbb{R}^{d\times d}$, $T \in \mathbb{N}$, $x_0 \in \mathcal{X}$, differentiable $f : \mathcal{X} \to \mathbb{R}^d$, $\phi > 0$
for $0 \le t < T$ **do**
  $x_{t+1} \leftarrow$ any (randomized) point in $\mathcal{X}$ satisfying

$$\mathbb{E}\left\langle \nabla f(x_t), x_{t+1}\right\rangle + \|x_{t+1} - x_t\|_{\mathbf{A}}^2 \le \min_{x\in\mathcal{X}} \left\{ \langle \nabla f(x_t), x\rangle + \|x - x_t\|_{\mathbf{A}}^2 \right\} + \frac{\phi}{20}$$

end
Return: x_T

Lemma 28 *Let $\phi > 0$, let $f : \mathcal{X} \to \mathbb{R}$ be twice-differentiable for convex $\mathcal{X} \subset \mathbb{R}^d$, and let $x^\star := \operatorname{argmin}_{x\in\mathcal{X}} f(x)$. Assume that $\frac{1}{2}\mathbf{A} \preceq \nabla^2 f(x) \preceq 2\mathbf{A}$ for all $x \in \mathcal{X}$, for positive definite $\mathbf{A} \in \mathbb{R}^{d\times d}$. Algorithm 3 with $T \leftarrow O(\log \frac{f(x_0)-f(x^\star)}{\phi})$ returns $x_T \in \mathcal{X}$ satisfying $\mathbb{E} f(x_T) \le f(x^\star) + \phi$.*

Proof Throughout the proof, let $\Phi_t := \mathbb{E} f(x_t) - f(x^\star)$, so our goal is to show $\Phi_T \le \phi$. We first observe that, conditioning on x_t, and letting $x^{(s)} := (1-s)x_t + sx^\star$,

$$\begin{aligned}
\mathbb{E} f(x_{t+1}) &\le \mathbb{E}\left[f(x_t) + \langle \nabla f(x_t), x_{t+1} - x_t\rangle + \|x_{t+1} - x_t\|_{\mathbf{A}}^2 \right] \\
&\le \min_{s\in[0,1]} \left\{ f(x_t) + \left\langle \nabla f(x_t), x^{(s)} - x_t \right\rangle + \left\| x^{(s)} - x_t \right\|_{\mathbf{A}}^2 \right\} + \frac{\phi}{20} \\
&\le \min_{s\in[0,1]} \left\{ f(x^{(s)}) + s^2 \|x_t - x^\star\|_{\mathbf{A}}^2 \right\} + \frac{\phi}{20} \\
&\le \min_{s\in[0,1]} \left\{ f(x^{(s)}) + 4s^2 \Phi_t \right\} + \frac{\phi}{20}.
\end{aligned}$$

Above, the first line used a second-order Taylor expansion and our assumption $\nabla^2 f(x) \preceq 2\mathbf{A}$ pointwise, the second line used the definition of x_{t+1}, the third used convexity, and the last used first-order optimality of $x^\star$ as well as our assumption $\frac{1}{2}\mathbf{A} \preceq \nabla^2 f(x)$ pointwise which implies that

$$\frac{1}{4}\left\|x_t - x^\star\right\|_{\mathbf{A}}^2 \le \langle \nabla f(x^\star), x_t - x^\star\rangle + \frac{1}{4}\left\|x_t - x^\star\right\|_{\mathbf{A}}^2 \le \Phi_t.$$

Subtracting $f(x^\star)$ from both sides and using convexity once more yields

$$\mathbb{E}\Phi_{t+1} \le \min_{s\in[0,1]}\left\{(1-s)\Phi_t + 4s^2\Phi_t\right\} + \frac{\phi}{20} = \frac{15}{16}\Phi_t + \frac{\phi}{20}.$$

Recursively applying for T iterations, and using $\frac{1}{20}\sum_{i=0}^{\infty}(\frac{15}{16})^i \le 1$, yields the conclusion. ■

Lemma 28 and Corollary 10 show that to implement a ball optimization oracle for the function $f = f_{\rho,\lambda,\bar{x}}$ (defined in (1)) over sufficiently small radii, it suffices to implement Line 3 of Algorithm 3 logarithmically many times. Concretely, when $\mathcal{X} = \mathbb{B}(r)$ and $f = f_{\rho,\lambda,\bar{x}}$, Line 3 requires a $\Theta(\phi)$-approximate minimizer to a problem of the form in (20), for $z \in \mathbb{B}(r)$ given by the algorithm. These are exactly the problems which our tools in Section A can approximately solve, except they are hard-constrained. We show how to lift the constraints via a regularized binary search in Section B.2.

B.2. Ball optimization oracles via binary search

In this section, we provide a binary search strategy for approximately solving the constrained optimization problem (20), by binary searching on a Lagrange multiplier for the constraint. To begin, we require the following claims on the minima of regularized convex functions.

Lemma 29 *Let $F : \mathbb{R}^d \to \mathbb{R}$ be a twice-differentiable convex function satisfying $\|\nabla F(\mathbb{0}_d)\| \le L$, and for all $\alpha \in \mathbb{R}_{\ge 0}$, let $x_\alpha^\star := \operatorname{argmin}_{x\in\mathbb{R}^d} F(x) + \frac{\alpha}{2}\|x\|^2$. We have the following claims.*

1. *For all $0 < \alpha < \alpha'$, $\|x_\alpha^\star\| > \|x_{\alpha'}^\star\|$.*
2. *For all $0 < \alpha < \alpha'$, $\|x_\alpha^\star - x_{\alpha'}^\star\| \le \|x_\alpha^\star\| \log(\frac{\alpha'}{\alpha})$.*
3. *If $\alpha \ge \frac{4L}{r}$, $\|x_\alpha^\star\| \le \frac{r}{2}$.*

Proof The optimality conditions on $x_\alpha^\star$ show that $\nabla F(x_\alpha^\star) = -\alpha x_\alpha^\star$, so differentiating in α,

$$\nabla^2 F(x_\alpha^\star)\left(\frac{\mathrm{d}}{\mathrm{d}\alpha}x_\alpha^\star\right) = -x_\alpha^\star - \alpha\cdot\frac{\mathrm{d}}{\mathrm{d}\alpha}x_\alpha^\star \implies \frac{\mathrm{d}}{\mathrm{d}\alpha}x_\alpha^\star = -\left(\nabla^2 F(x_\alpha^\star) + \alpha\mathbf{I}_d\right)^{-1}x_\alpha^\star.$$

Therefore, for any $0 < \alpha < \alpha'$, we have Item 1, as convexity of F shows

$$\frac{1}{2}\|x_{\alpha'}^\star\|^2 - \frac{1}{2}\|x_\alpha^\star\|^2 = \int_\alpha^{\alpha'}\left(-\|x_t^\star\|_{(\nabla^2 F(x_t^\star)+t\mathbf{I}_d)^{-1}}\right)\mathrm{d}t \le 0.$$

Now, by using the triangle inequality and Item 1, Item 2 follows:

$$\|x_\alpha^\star - x_{\alpha'}^\star\| \le \int_\alpha^{\alpha'}\left\|(\nabla^2 F(x_t^\star) + t\mathbf{I}_d)^{-1}x_t^\star\right\|\mathrm{d}t \le \int_\alpha^{\alpha'}\frac{1}{t}\|x_t^\star\|\,\mathrm{d}t \le \|x_\alpha^\star\|\log\left(\frac{\alpha'}{\alpha}\right).$$

Finally, to see Item 3, note that for $\alpha \geq \frac{4L}{r}$,

$$F(\mathbb{0}_d) \geq F(x^\star_\alpha) + \frac{\alpha}{2}\left\|x^\star_\alpha\right\|^2 \geq F(\mathbb{0}_d) + \langle \nabla F(\mathbb{0}_d), x^\star_\alpha\rangle + \frac{\alpha}{2}\left\|x^\star_\alpha\right\|^2 \geq F(\mathbb{0}_d) - L\left\|x^\star_\alpha\right\| + \frac{\alpha}{2}\left\|x^\star_\alpha\right\|^2.$$

Rearranging and applying our lower bound on α yields $\|x^\star_\alpha\| \leq \frac{r}{2}$ as claimed. ■

In light of Lemma 29, in the remainder of the section we fix a differentiable convex function F, and develop a generic framework for approximately solving, for a parameter $\lambda > 0$,

$$\text{argmin}_{x\in\mathbb{B}(r)} F(x) + \lambda\left\|x\right\|^2.$$

We follow the notation (19) throughout for brevity, so the above minimizer is denoted $x^\star_{2\lambda}$. For convenience, for any $t \in [0, \|x^\star_{2\lambda}\|]$, we also use α_t to denote the unique value of $\alpha \in [2\lambda, \infty)$ such that $\|x^\star_{\alpha_t}\| = t$, where uniqueness and existence follows from Lemma 29 and $x^\star_\infty = \mathbb{0}_d$.

At the end of Section A.4, we gave implementations of a phase-one oracle and a phase-two oracle for F in (21) (see Lemmas 26 and 27). We now apply Definitions 23 and 24 to implement our binary search, stated formally in the following and with pseudocode provided in Algorithm 4.

Proposition 30 *Let F be a differentiable convex function satisfying $\|\nabla F(\mathbb{0}_d)\| \leq L$. Let $\lambda, \Delta, r \in \mathbb{R}_{>0}$ with $\Delta \leq \frac{\lambda r^2}{100}$. Algorithm 4 computes $x \in \mathbb{B}(r)$ satisfying*

$$F(x) + \lambda\left\|x\right\|^2 \leq \min_{\|y\|\leq r} F(y) + \lambda\left\|y\right\|^2 + \Delta.$$

Algorithm 4 makes at most $O(\log \frac{L}{\lambda r})$ calls to $\mathcal{O}_1$, and $O(\log \frac{Lr+\lambda r^2}{\Delta})$ calls to $\mathcal{O}_2$.

Proof We start with a correctness proof, and bound the number of oracle calls at the end. Because the specifications of $\mathcal{O}_1$ and $\mathcal{O}_2$ do not preclude returning different answers on multiple calls with the same α, throughout the proof to alleviate burdensome notation, we assume that if an oracle is called twice with the same α, it gives the same result (e.g., the result of the first call).

We begin by analyzing the first phase of the algorithm, starting from Line 4 and ending before Line 4. By the criterion in the while loop on algorithm 4, u satisfied $\|\mathcal{O}_1(u)\| \leq 2.5r$, so

$$\|\mathcal{O}_1(u) - x^\star_u\| \leq \frac{1}{100}(r + \|x^\star_u\|),$$

which implies that for the value of u after the while loop ends,

$$\|x^\star_u\| - 2.5r \leq \|x^\star_u\| - \|\mathcal{O}_1(u)\| \leq \frac{1}{100}(r + \|x^\star_u\|) \implies \|x^\star_u\| \leq \frac{100}{99}\cdot(2.51r) < 3r.$$

Next, we claim that at the conclusion of phase one, either $\alpha' = 2\lambda$ and $\mathcal{O}_1(2\lambda) \leq 2.5r$, or α' has

$$\|x^\star_{\alpha'}\| \in [2r, 3r]. \tag{22}$$

The first case is obvious from Line 4. Otherwise, for the values of ℓ, u on Line 4, we have $\ell \geq 2\lambda$ and $\|\mathcal{O}_1(u)\| \leq 2.5r < \|\mathcal{O}_1(\ell)\|$. When the while loop breaks on Line 4, we have $\|\mathcal{O}_1(m)\| \in [2.1r, 2.9r]$, which yields (22) (since $\alpha' = m$ in this case) due to the following derivations:

$$\|x^\star_m\| - 2.9r \leq \|x^\star_m - \mathcal{O}_1(m)\| \leq \frac{1}{100}(r + \|x^\star_m\|) \implies \|x^\star_m\| \leq \frac{100}{99}\cdot(2.91r) < 3r,$$
$$2.1r - \|x^\star_m\| \leq \|x^\star_m - \mathcal{O}_1(m)\| \leq \frac{1}{100}(r + \|x^\star_m\|) \implies \|x^\star_m\| \geq \frac{100}{101}\cdot(2.09r) > 2r.$$

Algorithm 4: BinarySearch$(\lambda, r, \Delta, L, \mathcal{O}_1, \mathcal{O}_2)$

Input: $\lambda, r, \Delta, L \in \mathbb{R}_{>0}$, $\mathcal{O}_1$, an $(r, 2\lambda)$-phase-one oracle (Definition 23) for differentiable convex $f : \mathbb{R}^d \to \mathbb{R}$ satisfying $\|f(\mathbb{0}_d)\| \le L$, $\mathcal{O}_2$, a $(\frac{\Delta}{2}, r, \max(\alpha_{3r}, 2\lambda))$-phase-two oracle for f

// Start phase one.

$u \leftarrow 2\lambda$

while $\|\mathcal{O}_1(u)\| > 2.5r$ **do** $u \leftarrow 2u$

if $u = 2\lambda$ **then** $\alpha' \leftarrow 2\lambda$

else

  $\ell \leftarrow \frac{u}{2}$

  while true do

    $m \leftarrow \sqrt{u\ell}$

    if $\|\mathcal{O}_1(m)\| \in [2.1r, 2.9r]$ **then** $\alpha' \leftarrow m$ and **break**

    else if $\|\mathcal{O}_1(m)\| > 2.9r$ **then** $\ell \leftarrow m$

    else $u \leftarrow m$

  end

// Start phase two.

$\ell \leftarrow \alpha'$

$u \leftarrow \frac{4L}{r} + 2\lambda$

while $\frac{u}{\ell} > 1 + \frac{\Delta}{10(Lr+\lambda r^2)}$ **do**

  $m \leftarrow \sqrt{u\ell}$

  if $\|\mathcal{O}_2(m)\| > r$ **then** $\ell \leftarrow m$

  else $u \leftarrow m$

end

$x_1 \leftarrow \mathcal{O}_1(\ell)$, $x_2 \leftarrow \mathcal{O}_2(u)$

if $\ell = 2\lambda$ **then Return:** x_1

Return: $x_{\text{out}} \leftarrow (1-t)x_1 + tx_2$, where $t \in [0, 1]$ is chosen so $\|x_{\text{out}}\| = r$

Both bounds used the triangle inequality. This concludes our correctness analysis of Phase 1.

We now analyze correctness of phase two. By Item 3 of Lemma 29, we have $\|\mathcal{O}_2(u)\| \le \|x_u^\star\| + \|x_u^\star - \mathcal{O}_2(u)\| \le r$ on Line 4, where we used strong convexity of $F(x) + \frac{\alpha}{2}\|x\|^2$ to bound $\|x_u^\star - \mathcal{O}(u)\| \le \frac{r}{2}$. Thus, inspecting the while loop starting on Line 4, we preserve the invariants:

$$\ell < u, \ \|\mathcal{O}_2(u)\| \le r, \text{ and either } \|\mathcal{O}_2(\ell)\| > r, \text{ or } \ell = 2\lambda.$$

In particular, if $\|\mathcal{O}_2(\ell)\| \le r$, it must be that $\alpha' = \ell$ (i.e. ℓ never updated), but if $\alpha' \ne 2\lambda$ then this is impossible by (22). Hence, when the while loop on algorithm 4 terminates, the values ℓ, u associated with x_1, x_2 satisfy $u \in [\ell, (1 + \frac{\Delta}{10(Lr+\lambda r^2)})\ell]$, $\|x_2\| \le r$, and we are in one of the following cases.

1. $\ell = 2\lambda$ and $\mathcal{O}_2(u) \le r$.
2. $x_1 = \mathcal{O}_2(\ell)$ has $\|x_1\| > r$.

In Case 1, let $y := \mathrm{argmin}_{y \in \mathbb{B}(r)} F(y) + \lambda \|y\|^2$. Then by the guarantees of $\mathcal{O}_2$,

$$F(x_1) + \lambda \|x_1\|^2 \le F(x_1) + \frac{u}{2}\|x_1\|^2 \le F(y) + \frac{u}{2}\|y\|^2 + \frac{\Delta}{2}$$
$$\le F(y) + \lambda \|y\|^2 + \left(\frac{u}{2} - \lambda\right) r^2 + \frac{\Delta}{2} \le F(y) + \lambda \|y\|^2 + \Delta,$$

where we used $\frac{u}{2} - \lambda \le \frac{\Delta}{10\lambda r^2} \cdot \frac{\lambda}{2} \le \frac{\Delta}{2r^2}$. On the other hand, in Case 2, recalling $\|x_\ell^\star\|, \|x_u^\star\| \le 3r$ by the guarantees of phase one, and letting $x_{\mathrm{out}} := (1-t)x_1 + tx_2$ as in Line 4,

$$(1-t)F(x_1) + tF(x_2) + \frac{(1-t)\ell}{2}\|x_1\|^2 + \frac{tu}{2}\|x_2\|^2 \le F(y) + \frac{u}{2}\|y\|^2 + \frac{\Delta}{2},$$

for every $y \in \mathbb{R}^d$, by the definition of $\mathcal{O}_2$. Now, letting $y := \mathrm{argmin}_{y \in \mathbb{B}(r)} F(y) + \lambda \|y\|^2$,

$$F(x_{\mathrm{out}}) + \frac{\ell}{2}\|x_{\mathrm{out}}\|^2 \le (1-t)F(x_1) + tF(x_2) + \frac{(1-t)\ell}{2}\|x_1\|^2 + \frac{tu}{2}\|x_2\|^2$$
$$\le F(y) + \frac{u}{2}\|y\|^2 + \frac{\Delta}{2}.$$

Additionally, note that if we are in Case 2, then $\|y\| = r$. To see this, suppose for contradiction that $\|y\| < r$, which means $\|x_{2\lambda}^\star\| < r$. If $\alpha' > 2\lambda$, then (22) and Item 1 of Lemma 29 give a contradiction. Otherwise, $\alpha' = 2\lambda$, but then $\|x_{2\lambda}^\star\| < r$ contradicts the statement before (22) since $\mathcal{O}_1(2\lambda) \le 2.5r$ cannot happen. Hence, $\|y\| = \|x_{\mathrm{out}}\| = r$, and correctness in Case 2 follows from

$$F(x_{\mathrm{out}}) + \lambda \|x_{\mathrm{out}}\|^2 = F(x_{\mathrm{out}}) + \lambda \|y\|^2$$
$$\le F(y) + \lambda \|y\|^2 + \frac{(u-\ell)r^2}{2} + \frac{\Delta}{2} \le F(y) + \lambda \|y\|^2 + \Delta,$$

where $u - \ell \le \frac{\Delta}{10(Lr + \lambda r^2)} \cdot u \le \frac{\Delta}{r^2}$, since $u \le \frac{4L}{r} + 2\lambda$. This completes the correctness proof.

We now bound the number of calls to $\mathcal{O}_1, \mathcal{O}_2$. By Item 3 of Lemma 29 and (22), it is clear the number of times Line 4 occurs is $O(\log \frac{L}{\lambda r})$. Next, consider the loop on Algorithm 4 until Algorithm 4 is hit. We claim the loop must break if $\log \frac{u}{\ell} \le \frac{1}{100}$, which means the loop can only run $O(1)$ times, because $\log \frac{u}{\ell}$ halves each time the loop is run, and $\frac{u}{\ell} = 2$ initially. To see our claim, for any α,

$$\begin{aligned} \|\mathcal{O}_1(\alpha)\| - \|x_\alpha^\star\| \le \frac{r + \|x_\alpha^\star\|}{100} &\implies \frac{100}{101}\|\mathcal{O}_1(\alpha)\| - \frac{r}{101} \le \|x_\alpha^\star\|, \\ \|x_\alpha^\star\| - \|\mathcal{O}_1(\alpha)\| \le \frac{r + \|x_\alpha^\star\|}{100} &\implies \frac{100}{99}\|\mathcal{O}_1(\alpha)\| + \frac{r}{99} \ge \|x_\alpha^\star\|, \end{aligned} \tag{23}$$

which follow from the definition of $\mathcal{O}_1$. Further, by Item 2 of Lemma 29, supposing $\log \frac{u}{\ell} \le \frac{1}{100}$,

$$\|x_u^\star - x_\ell^\star\| \le \frac{\|x_\ell^\star\|}{100} \le \frac{1}{100}\left(\frac{100}{99}\|\mathcal{O}_1(\ell)\| + \frac{r}{99}\right),$$

where we used the second bound in (23). Combining with (23), we have

$$\frac{1}{100}\left(\frac{100}{99}\|\mathcal{O}_1(\ell)\| + \frac{r}{99}\right) \ge \|x_\ell^\star\| - \|x_u^\star\| \ge \left(\frac{100}{101}\|\mathcal{O}_1(\ell)\| - \frac{r}{101}\right) - \left(\frac{100}{99}\|\mathcal{O}_1(u)\| + \frac{r}{99}\right)$$
$$\implies \frac{100}{99}\|\mathcal{O}_1(u)\| + \frac{r}{33} \ge \left(\frac{100}{101} - \frac{1}{99}\right)\|\mathcal{O}_1(\ell)\|,$$

which is a contradiction since $\|\mathcal{O}_1(u)\| < 2.1r$ and $\|\mathcal{O}_1(\ell)\| > 2.9r$ until termination.

Finally, consider the loop starting on Line 4. At the beginning, we have $\frac{u}{\ell} = O(\frac{L}{\lambda r} + 1)$, and $\log \frac{u}{\ell}$ halves each time the loop is run. Therefore, $\mathcal{O}_2$ is called $O(\frac{Lr+\lambda r^2}{\Delta})$ times as claimed. ■

We now combine Proposition 30 with Lemmas 26 and 27 to give our parallel ball optimization oracle.

Proposition 31 *Define f_ρ as in Definition 5, where f is in the setting of Problem 1. Let $\lambda, r \in \mathbb{R}_{>0}$ satisfy $r \le \frac{\rho}{6} \cdot \log^{-\frac{1}{2}}(\frac{2L}{\lambda\rho})$ and $\rho \le \frac{L}{\lambda}$. For any $\phi \in (0, \frac{\lambda r^2}{100}]$, we can implement a (ϕ, λ, r)-ball optimization oracle (Definition 6) for f_ρ with*

$$O\left(\log\left(\frac{Lr}{\phi}\right)\log\log\left(\frac{Lr}{\phi}\right)\cdot\mathcal{D}_{\text{query}} + \frac{\lambda r^2}{\phi}\log^4\left(\frac{L^2}{\lambda\phi}\right)\log\left(\frac{dL^2}{\lambda\phi}\right)\right) \text{ depth,}$$

$$\text{and } O\left(\frac{L^2}{\lambda\phi}\log^5\left(\frac{L^2}{\lambda\phi}\right)\cdot\mathcal{T}_{\text{query}} + d\log^5\left(\frac{L^2}{\lambda\phi}\right)\cdot\frac{\lambda r^2}{\phi}\cdot\left(\frac{L^2}{\lambda^2 r^2}\right)^{\omega-1}\right) \text{ work.}$$

Proof Throughout, assume $\bar{x} = \mathbb{0}_d$ in the definition of the ball optimization oracle, which is without loss of generality because shifting by a constant vector does not affect the assumptions in Problem 1. Also, define $f_{\rho,\lambda,\bar{x}}$ as in (1), where $\bar{x} = \mathbb{0}_d$. We give an algorithm which always returns a point x in $\mathbb{B}(r)$, and such that x has suboptimality gap $\frac{\phi}{2}$, except with probability $\delta := \frac{\phi}{2Lr+\lambda r^2}$. Because f is L-Lipschitz by Jensen's inequality on the moment bound in Problem 1, so is f_ρ by Fact 1. Therefore the range of $f_{\rho,\lambda,\bar{x}}$ over $\mathbb{B}(r)$ is at most $Lr + \lambda r^2$, and the expected suboptimality gap is

$$(1-\delta)\cdot\frac{\phi}{2} + \delta\cdot\left(Lr + \frac{\lambda r^2}{2}\right) \le \phi,$$

as required. To implement this algorithm, we first apply Lemma 28 and Corollary 10, which show that it suffices to solve $T = O(\log\frac{Lr+\lambda r^2}{\phi}) = O(\log\frac{Lr}{\phi})$ problems of the form, for some $z \in \mathbb{B}(r)$,

$$\underbrace{\langle\nabla f_\rho(z) - \lambda z, x\rangle + \|x - z\|^2_{\nabla^2 f_\rho(\mathbb{0}_d)}}_{:=F(x)} + \lambda\|x\|^2,$$

each to error $\Delta := \frac{\phi}{40}$ (see (20) for the derivation). Note that

$$\|\nabla F(\mathbb{0}_d)\| = \left\|\nabla f_\rho(z) - \lambda z - 2\nabla^2 f_\rho(\mathbb{0}_d)z\right\| \le 2L + \lambda r,$$

because f_ρ is L-Lipschitz and $\frac{L}{\rho}$-smooth (Fact 1). Finally, let $Z := O(\log\frac{Lr}{\phi})$ be the total number of oracle calls to $\mathcal{O}_1$ or $\mathcal{O}_2$ used by Proposition 30. We implement each oracle using either Lemma 26 or Lemma 27 appropriately, with failure probability set to $\frac{\delta}{Z}$, and the conclusion follows. ■

We also note that we can achieve a computational depth-complexity tradeoff in Proposition 11 by choosing different values of S in Corollary 21, than was used in Lemma 27. As stated, Lemma 27 uses Corollary 21 by applying our parallel implementation S iterations at a time, where $S = \frac{L^2}{\alpha\lambda r^2}$ is the number of iterations required to achieve $\approx \lambda r^2$ error. By instead choosing a larger error $C \cdot \lambda r^2$ for a parameter $C \in [1, \frac{L^2}{\lambda^2 r^2}]$, which induces $S = \frac{L^2}{C\alpha\lambda r^2}$, we can obtain improved total work bounds at the cost of larger computational depth. In particular, Corollary 21 has a computational depth scaling linearly in the parameter C, and a computational complexity scaling as $C^{2-\omega}$; all logarithmic terms are unnaffected, since $C \le \frac{L^2}{\lambda^2 r^2}$. We summarize this observation in the following.

Corollary 32 *In the context of Proposition 11, for any $C \in [1, \frac{L^2}{\lambda^2 r^2}]$, we can implement a (ϕ, λ, r)-ball optimization oracle (Definition 6) for f_ρ with*

$$O\left(\log\left(\frac{Lr}{\phi}\right)\log\log\left(\frac{Lr}{\phi}\right)\cdot\mathcal{D}_{\text{query}} + \frac{C\lambda r^2}{\phi}\log^4\left(\frac{L^2}{\lambda\phi}\right)\log\left(\frac{dL^2}{\lambda\phi}\right)\right) \text{ depth,}$$

$$\text{and } O\left(\frac{L^2}{\lambda\phi}\log^5\left(\frac{L^2}{\lambda\phi}\right)\cdot\mathcal{T}_{\text{query}} + d\log^5\left(\frac{L^2}{\lambda\phi}\right)\cdot\frac{\lambda r^2}{\phi}\cdot\left(\frac{L^2}{\lambda^2 r^2}\right)^{\omega-1}\cdot\left(\frac{1}{C}\right)^{\omega-2}\right) \text{ work.}$$