ON DISTRIBUTED SEPARATION ORACLES AND THE COMMUNICATION COST OF OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a new model for studying distributed optimization, that we call the distributed separation oracle model. Here one is allowed to make separation oracle queries to one of s constraint sets $\mathcal{C}_i$ at a time. In this model we show that one can solve linear programming with $O(s + L \log s)$ queries, for constant d, where L represents the bit precision to which the underlying constraints are specified. This improves on the naïve approach of making s queries per round to simulate separation oracle queries to $\bigcap \mathcal{C}_i$, and suggests that further non-trivial algorithms may be possible in this model. We also show that a similar improvement is possible in constant dimensions, even for adversarial separation oracles, with no bit-precision limits on the query results.

We also study the problem of solving linear programs in a distributed setting with many servers. In the blackboard and coordinator models of communication, we develop algorithms for linear programming, and related convex optimization problems that improve over the previous best of [Vempala, Woodruff, Wang '20]. Specifically, we give an algorithm using $O(sd^3L + \text{poly}(d \log n) \cdot L\sqrt{s}$ communication in the coordinator model with s servers, and $O(sd) + (L \log n + \log^2 n + \sqrt{s}) \log(sd) \cdot \text{poly}(d)$ communication in the blackboard model.

1 INTRODUCTION

Convex optimization forms the mathematical engine powering a vast array of machine learning paradigms. Prominent examples include robust ℓ_p regression for outlier rejection, semi-definite programming (SDP) for metric learning and matrix completion, and the training of models subject to strict fairness, safety, or robustness guarantees Vandenberghe & Boyd (1996). In modern large-scale settings such as cross-device federated learning and multi-agent systems, these problems are often inherently distributed across devices, institutions, or edge systems, where data cannot be centralized due to privacy, storage, or bandwidth constraints. Consequently, the primary bottleneck in distributed machine learning is communication overhead. This has motivated the design of scalable algorithms that achieve near-optimal solutions while minimizing communication complexity.

Historically, the machine learning and optimization communities have relied on the elegant abstraction of the *separation oracle* to handle complex constraints Yudin & Nemirovskii (1976); Dadush et al. (2024); Grötschel et al. (2012). Given a candidate point x, a separation oracle either certifies feasibility or returns a hyperplane separating x from the feasible region, naturally underlying cutting-plane methods in structured prediction, e.g., identifying most-violated constraints in structured SVMs, as well as adversarial training, where one seeks worst-case violations or perturbations. In distributed settings, however, constraints are inherently partitioned across machines holding local data, and explicitly transmitting them is infeasible. For example, a polytope defined by n local points in d dimensions can have $\mathcal{O}(n^{\lfloor d/2 \rfloor})$ facets Gale (1963), which is prohibitively expensive. This motivates communication-efficient methods that simulate separation oracles for distributed datasets.

To formalize this communication barrier in a decentralized optimization setting, we introduce the *Distributed Separation Oracle (DSO)* model. In this framework, a collection of convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$, each representing feasibility constraints induced by the local data of a client, is distributed across s servers. A query consists of a point x and a chosen server index i, and returns a separation oracle response with respect to $\mathcal{C}_i$. Simulating a global separation oracle for the intersection $\bigcap_{i=1}^{s} \mathcal{C}_i$ thus naïvely requires interacting with all s servers per round. This raises a basic

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

question in distributed optimization: *can one solve convex programs in the DSO model with strictly fewer than s oracle queries per iteration compared to the centralized separation oracle model?*

1.1 Our Contributions

We answer this question in the affirmative, showing that convex programs in the Distributed Separation Oracle (DSO) model can indeed be solved with strictly fewer than s oracle queries per iteration. More broadly, we develop new algorithms and lower bounds for decentralized optimization under two complementary settings: the DSO model, where interaction is via oracle queries, and classical distributed models, where efficiency is measured in communication.

The distributed separation oracle model and query complexity. We begin with the DSO model introduced above, where each server holds a convex set $\mathcal{C}_i$ and interaction is restricted to separation oracle queries. Our objective is to minimize a known linear functional $\phi(x) = \langle c, x\rangle$, where $c \in \mathbb{R}^d$.

A direct implementation of a cutting-plane method queries all s servers at each iteration. To reduce the volume of a full-dimensional localization region to an η fraction of its initial volume, where $0 < \eta < 1/2$, the center-of-gravity and ellipsoid methods use $O(\log(1/\eta))$ and $O(d\log(1/\eta))$ iterations, respectively Grünbaum (1960); Vempala et al. (2020). Querying every server therefore gives $O(s\log(1/\eta))$ and $O(sd\log(1/\eta))$ oracle queries, respectively. Our goal is to reduce this multiplicative dependence on s.

In the bit-limited (BL-DSO) setting, the input inequalities, objective, and oracle-returned separating hyperplanes have L-bit coefficients. For our exact-optimization upper bounds, we assume that the feasible region $\bigcap_{i=1}^{s}\mathcal{C}_i$ is non-empty and contained in the public box $Q = [-1,1]^d$. Applying our exact cutting-plane subroutine directly to all servers gives an $O(sL\cdot\mathrm{poly}(d))$ query baseline, or $O(sL)$ for constant dimension. We improve this dependence on s and L, without any dependence on the number of inequalities defining the underlying convex sets.

Theorem 1.1. *Let $\mathcal{C}_1,\dots,\mathcal{C}_s \subseteq \mathbb{R}^d$ be constraint sets in the BL-DSO model, with non-empty intersection $\mathcal{C} = \bigcap_{i=1}^{s}\mathcal{C}_i \subseteq [-1,1]^d$. Let $\phi(x) = \langle c, x\rangle$ be a known linear functional with L-bit coefficients. There is a randomized algorithm that returns an exact minimizer $x_* \in \mathcal{C}$ of ϕ, using $O(sd) + L\sqrt{s}\cdot\mathrm{poly}(d)$ separation oracle queries in expectation.*

For constant dimension, we further improve the $O(sL)$ baseline to $O(s + L\log s)$ queries. Together with a lower bound of $\Omega(s+L)$ queries even for feasibility, this establishes near-optimality up to a logarithmic factor.

Theorem 1.2. *Let d be constant, and let $\mathcal{C}_1,\dots,\mathcal{C}_s \subseteq \mathbb{R}^d$ be constraint sets in the BL-DSO model, with non-empty intersection $\mathcal{C} = \bigcap_{i=1}^{s}\mathcal{C}_i \subseteq [-1,1]^d$. For any known linear functional $\phi(x) = \langle c, x\rangle$ with L-bit coefficients, there is a randomized algorithm that returns an exact minimizer of ϕ over $\mathcal{C}$ using $O(s + L\log s)$ separation oracle queries in expectation. Moreover an algorithm that solves the convex feasibility problem must use at least $\Omega(s+L)$ separation oracle queries.*

We also study a more general setting without bit-precision assumptions, where exact optimization is no longer directly accessible. In this case, we consider the feasibility problem under a mild volume assumption and show that sublinear dependence on s per round is still achievable.

Theorem 1.3. *Let $\mathcal{C}_1,\dots,\mathcal{C}_s$ be convex sets contained in the unit ball of $\mathbb{R}^d$, each accessible via separation oracle queries. Let $\mathcal{C} = \cap_i\mathcal{C}_i$ and suppose that $\mathrm{Vol}(\mathcal{C}) \geq \varepsilon$. There is an algorithm which outputs a point $x_* \in \mathcal{C}$ with probability at least $3/4$ and makes at most $O\left(\frac{s}{\alpha} + s^{\alpha}\log s\log\frac{1}{\varepsilon}\right)$ queries to the separation oracles. Here $\alpha > 0$ can be taken to be any value, and c_d depends only on d and α.*

Explicit communication models and bit complexity. We next study classical distributed settings in which each of s servers holds a convex set $\mathcal{C}_i$, typically given as the intersection of local linear constraints, and the goal is to optimize a linear functional ϕ while minimizing total communication. We focus on two standard models: the coordinator model and the blackboard model. Prior work achieves communication bounds that scale multiplicatively with both the number of servers s and the total number of constraints n, which becomes prohibitive in large-scale settings. We give new algorithms that break this multiplicative dependence, decoupling the roles of s and n and yielding strictly improved bounds. In the coordinator model, we obtain the following result.

Theorem 1.4. *There is an algorithm that solves distributed linear programming using total communication $O(sd^3L)+\mathrm{poly}(d\log n)\cdot L\sqrt{s}$ in the coordinator model, where each $\mathcal{C}_i$ is an intersection of n_i half-spaces expressed to L bits of precision, $n=\sum n_i$, and ϕ is expressed to L bits of precision.*

For large s and n, this improves on Vempala et al. (2020), which achieves $O(sd^3L\log n)$ communication. We also obtain an analogous improvement in the blackboard model, where we reduce the leading term from $sd\log n$ to sd.

Theorem 1.5. *There is an algorithm that solves distributed linear programming using total communication $O(sd)+(L\log n+\log^2 n+\sqrt{s})\log(sd)\cdot\mathrm{poly}(d)$ in the blackboard model, where each $\mathcal{C}_i$ is an intersection of n_i half-spaces expressed to L bits of precision, $n=\sum n_i$, and ϕ is expressed to L bits of precision.*

In particular, for constant d, Theorem 1.5 uses $O(s)+(L\log n+\log^2 n+\sqrt{s})\log(s)$ communication. We summarize these results and their improvements over previous work in Table 1.

Table 1: Communication complexity for distributed linear programming with s servers, d dimensions, n total local constraints, and L bits of precision. Prior results in Vempala et al. (2020) exhibit a multiplicative penalty between s and $\log n$ in their leading terms. Our protocols successfully decouple the local dataset size (n) from the network scale (s).

Model	**Previous Best**	**Our Work**
Coordinator	$O(sd^3L\log n)$	$O(sd^3L)+\mathrm{poly}(d\log n)L\sqrt{s}$
Blackboard	$\tilde{O}(sd\log n)$	$O(sd)+(L\log n+\log^2 n+\sqrt{s})\log(sd)\cdot\mathrm{poly}(d)$

Our approach also extends to settings with an additional closed convex constraint $\mathcal{G}$ known to all parties. We retain $\mathcal{G}\cap[-1,1]^d$ in every optimization subproblem, under the optimizer-containment and exact local-computation assumptions stated below. Even if the resulting x_* cannot be expressed with few bits, it can be communicated implicitly by sending at most d input inequalities whose lexicographically smallest optimum over $\mathcal{G}\cap[-1,1]^d$ is x_*. Under these assumptions, our communication bounds also apply to semidefinite programming Vandenberghe & Boyd (1996); Gärtner & Welzl (1996)[1].

Theorem 1.6. *Suppose each of s servers holds a convex set $\mathcal{C}_i$, which is an intersection of n_i half spaces expressed to L bits of precision, and wish to optimize a linear functional ϕ expressed to L bits of precision. The communication guarantees of Theorem 1.4 and Theorem G.3 hold under the addition of a global convex constraint $\mathcal{G}$. That is, a server outputs x_* with $\phi(x_*)$ minimal in $\mathcal{G}\cap(\bigcap_i\mathcal{C}_i)$.*

While this is a straightforward modification of the above results, we point it out explicitly, since this formulation includes semi-definite programs as a special case.

Applications to distributed ℓ_p regression. We apply our communication bounds to distributed ℓ_p regression for fixed constant $p>2$ via a standard reduction. Prior work Vempala et al. (2020) shows that computing a $(1+\varepsilon)$-approximate ℓ_p regression solution can be reduced to solving a linear program of dimension $\tilde{O}(d/\varepsilon^2)$ with $\tilde{O}(nd/\varepsilon^2)$ constraints. Accounting for the resulting coefficient precision and applying our improved linear programming algorithms gives the following result.

Theorem 1.7. *Fix a constant $p>2$ and let $0<\varepsilon<1/2$. Given $A\in\mathbb{R}^{n\times d}$ and $b\in\mathbb{R}^n$ with L-bit entries distributed by rows among s servers, there is an algorithm that with constant probability finds $\hat{x}$ satisfying*

$$\|A\hat{x}-b\|_p\le(1+\varepsilon)\cdot\min_x\|Ax-b\|_p.$$

The algorithm uses expected communication $\tilde{O}\left(s(d/\varepsilon^2)^3L+L\sqrt{s}\cdot\mathrm{poly}(d/\varepsilon)\right)$ in the coordinator model and $\tilde{O}\left(sd/\varepsilon^2+(d/\varepsilon^2)^{O(1+\log\log(s+2))}L\right)$ in the blackboard model. Here $\tilde{O}$ suppresses factors polylogarithmic in n, d, and $1/\varepsilon$, and hidden constants may depend on p.

[1]Semidefinite programming optimizes a linear functional of a symmetric matrix subject to linear constraints and the requirement that the matrix be positive semidefinite.

In particular, the Clarkson-based blackboard bound Vempala et al. (2020) incurs an $s \log n$ dependence. Our linear programming protocols remove this $\log n$ factor from the term linear in s; the explicit parameter dependence is given in the proof.

1.2 Technical Overview: Contextualizing Our Contributions

A major bottleneck in distributed convex optimization is that standard algorithms typically require full-network coordination at every iteration. For instance, distributed adaptations of Clarkson's algorithm Clarkson (1995); Vempala et al. (2020) require maintaining and updating weights across the entire global set of constraints, while classical cutting-plane methods Yudin & Nemirovskii (1976); Grötschel et al. (2012) require polling every server to find violated constraints. Because identifying a single progress-inducing constraint involves aggregating information across all s servers, the communication and query costs scale multiplicatively with both the network size s and the total number of constraints n. This full-system dependence is precisely what drives the $\tilde{\mathcal{O}}(sd \log n)$ communication cost in the prior state-of-the-art Vempala et al. (2020), and enforces a strict linear $\Omega(s)$ query lower bound per iteration in oracle models.

Our approach bypasses this bottleneck by exploiting a fundamental property of convex geometry: regardless of the number of servers s or constraints n, the global optimum is completely determined by a small active basis of at most d constraints. Instead of repeatedly reaching a consensus across the entire network, we reframe the optimization as a targeted search for this active set. By focusing on discovering these critical constraints, we safely decouple the network scale from the dataset size. We execute this through four primary technical mechanisms, which overcome specific limitations of prior approaches:

Bounding global coordination via active-set sampling. Previous distributed linear programming methods Vempala et al. (2020) implicitly assume that making algorithmic progress requires evaluating the entire dataset. We instead restrict the optimization to dynamically sampled, local subsystems of only $\mathcal{O}(d\sqrt{s})$ servers. Because the true optimal basis contains at most d constraints, any candidate solution that is optimal for the sampled subset but globally infeasible must violate at least one previously unseen basis constraint. This structural guarantee ensures that our algorithm only needs $\mathcal{O}(d)$ global coordination rounds to find the true basis, completely removing the dependence on the total number of constraints n from the outer optimization loop.

Exact optimization from purely geometric queries. Applying this sampling framework to the Bit-Limited Distributed Separation Oracle (BL-DSO) model introduces a new challenge: recovering an exact optimizer using only separation-oracle responses, since servers cannot explicitly transmit constraints. Classical exact solvers Gleixner & Steffy (2020); Dadush et al. (2024) eventually require explicitly extracting constraints to pinpoint the optimal vertex. On the other hand, purely geometric methods like the Center-of-Gravity algorithm only shrink the search space asymptotically, meaning they cannot isolate an exact optimum in finite time. To obtain an exact solution, we introduce a combinatorial step based on Diophantine approximation. Let $\Delta_{L,d} \subseteq \mathbb{R}^d$ denote the finite set of points that arise as unique intersections of d hyperplanes whose normal vectors and offsets have L-bit coefficients. Every non-empty subproblem, restricted to the public box, has an optimal vertex in $\Delta_{L,d}$. A lower bound on the volumes of nondegenerate simplices with vertices in $\Delta_{L,d}$ implies that, once the current localization region W has sufficiently small intrinsic volume, the remaining candidates $W \cap \Delta_{L,d}$ lie in a lower-dimensional affine space. Our algorithm explicitly restricts W to the convex hull of these candidates and continues within its affine hull, preserving an optimal vertex until an exact optimizer is isolated.

Sublinear queries via randomized exploration. In general continuous settings without bit-precision limits, finding a violated constraint typically requires a deterministic sweep over all s servers—otherwise, an adversarial oracle could hide the violation on the last unqueried server, enforcing a strict $\Omega(s)$ query cost per step. We circumvent this worst-case behavior by treating constraint discovery as an online randomized hidden-structure game (which we abstract as a "light bulb" problem). By maintaining a probability distribution over the feasible region and sampling query points randomly rather than deterministically, we can reliably flush out active violations without exhaustive polling. This allows us to achieve strictly sublinear queries per iteration, even against adversarial oracles.

Decoupling scale via nested multiplicative weights. Finally, when implementing our active-set framework in explicit communication models (the Blackboard and Coordinator settings), using standard Multiplicative Weights Updates (MWU) over constraints—the core engine of Vempala et al. (2020)—would reintroduce an $\mathcal{O}(\log n)$ communication penalty globally. To eliminate this coupling, we design a *Nested Multiplicative Weights* (NESTEDMWU) subroutine. Rather than maintaining global weights, NESTEDMWU confines the reweighting entirely to the sampled subsystem of $\mathcal{O}(d\sqrt{s})$ servers. It aggressively amplifies the weights of locally violated constraints to isolate the true geometric basis in merely $\mathcal{O}(d\log(sd))$ local steps. This nested isolation successfully factors the local dataset scale, represented by $\log n$, out of the leading network term, $\mathcal{O}(sd)$, yielding our strictly improved communication bounds.

2 A UNIFIED ACTIVE-SET FRAMEWORK FOR DISTRIBUTED GEOMETRY

At the heart of our results across all distributed models is a structural viewpoint: the fundamental difficulty of distributed optimization is not the scale of the global constraint set, but the challenge of identifying a dynamically changing, low-dimensional subset of critical constraints. Prior decentralized methods repeatedly simulate full-system constraint aggregation, thereby forcing each iteration to interact with all s servers Clarkson (1995); Vempala et al. (2020). We bypass this global synchronization barrier by exploiting a hidden active-set geometry: regardless of the number of servers s or local constraints n, the selected global optimizer is determined by constraints held on at most d servers, together with the public domain constraints. We formalize this via two core geometric lemmas. We state these lemmas for optimization over a fixed compact convex public domain Q, taking $Q = [-1, 1]^d$ in the BL-DSO setting. To resolve ties consistently, we select the lexicographically smallest minimizer. First, we show that this selected optimizer is supported by a global basis of at most d servers.

Lemma 2.1. *Let $Q \subseteq \mathbb{R}^d$ be a compact convex set, and let $\mathcal{S}$ be a finite family of closed convex sets such that $Q \cap \bigcap_{\mathcal{C}\in\mathcal{S}} \mathcal{C} \neq \emptyset$. Let ϕ be a linear functional. For each $\mathcal{A} \subseteq \mathcal{S}$, let $x_{\mathcal{A}}$ be the lexicographically smallest minimizer of ϕ over*

$$P_{\mathcal{A}} := Q \cap \bigcap_{\mathcal{C}\in\mathcal{A}} \mathcal{C}, \qquad P_{\emptyset} := Q.$$

There exists $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ with $|\tilde{\mathcal{S}}| \leq d$ such that $x_{\tilde{\mathcal{S}}} = x_{\mathcal{S}}$.

The intuition for Lemma 2.1 follows from Helly's theorem. Order points by increasing objective value, breaking ties lexicographically by their coordinates. Consider the convex set of points that precede $x_{\mathcal{S}}$ in this order. Its intersection with Q and all sets in $\mathcal{S}$ is empty. Since $x_{\mathcal{S}}$ belongs to Q and every set in $\mathcal{S}$, a Helly witness must include the set of preceding points and at most d other sets. Retaining Q and the server sets in this witness preserves $x_{\mathcal{S}}$ while excluding every preceding point, translating to a basis of at most d servers.

Our second core geometric lemma claims that if we optimize over a sufficiently large random subset of servers, the resulting candidate optimum will violate only a small fraction of the remaining unobserved constraints.

Lemma 2.2. *With the notation and assumptions of Lemma 2.1, fix $\mathcal{T} \subseteq \mathcal{S}$ and choose $\mathcal{R}$ uniformly from $\binom{\mathcal{S}}{r}$, where $1 \leq r \leq |\mathcal{S}|$. For $\mathcal{V}_{\mathcal{T},\mathcal{R}} = \{\mathcal{C} \in \mathcal{S} : x_{\mathcal{T}\cup\mathcal{R}} \notin \mathcal{C}\}$, we have $\mathbb{E}[|\mathcal{V}_{\mathcal{T},\mathcal{R}}|] \leq d|\mathcal{S}|/r$.*

Together, Lemma 2.1 and Lemma 2.2 give rise to a model-agnostic *dynamic active-server* meta-algorithm. Instead of continually querying all s servers, the algorithm proceeds by progressively uncovering the global basis $\mathcal{B}^*$ without ever fully aggregating the system of constraints. Informally, the meta-algorithm maintains an active set of known violating servers $\mathcal{T}$. In each epoch, it executes the following steps:

1. **Subproblem Generation:** Draw a uniformly random subset of servers $\mathcal{R} \subseteq [s]$ of size $\lceil d\sqrt{s}\rceil$.
2. **Local Optimization:** Compute a candidate optimum x_* of ϕ over $Q \cap \bigcap_{i\in\mathcal{T}\cup R} \mathcal{C}_i$.
3. **Violation Discovery:** Check x_* against the full network to find the set of all violated servers $\mathcal{V}$.
4. **Active Set Expansion:** If $|\mathcal{V}| \leq 2\sqrt{s}$, add $\mathcal{V}$ to $\mathcal{T}$. Otherwise, discard the epoch. Repeat until $\mathcal{V} = \emptyset$.

Conditioned on the preceding rounds, T is fixed and $\mathcal{R}$ is a fresh uniform sample. Lemma 2.2 therefore gives $\mathbb{E}[|V| \mid T] \leq \sqrt{s}$. By Markov's inequality, each iteration satisfies $|V| \leq 2\sqrt{s}$ with probability at least $\frac{1}{2}$. Moreover, any globally infeasible candidate must violate a member of a fixed global basis $\mathcal{B}^*$ outside $\mathcal{T}$. Each successful intermediate iteration therefore adds at least one new basis member to $\mathcal{T}$. Since $|\mathcal{B}^*| \leq d$, there are at most d such iterations. Thus we have:

Lemma 2.3. *The active-server meta-algorithm terminates almost surely and returns the lexicographically smallest minimizer x_* of ϕ over the full feasible region. It performs at most $d + 1$ successful iterations and $O(d)$ total iterations in expectation.*

This establishes a powerful **global-to-local reduction**: any distributed convex program over s servers can be exactly solved by resolving just $O(d)$ restricted subproblems, each involving a dynamically chosen subsystem of at most $O(d\sqrt{s})$ servers. The remainder of this paper is dedicated to instantiating this meta-algorithm in different environments. The primary challenge lies in how to efficiently compute the local optimum on $\mathcal{T} \cup \mathcal{R}$ (Step 2) and how to identify global violations (Step 3). As we will show, implementing these steps via geometric oracle queries yields the first strictly sublinear results in the implicit Distributed Separation Oracle (DSO) model (Section 3), while implementing them via explicit message-passing yields state-of-the-art bounds for **communication complexity** (Section 5).

3 Distributed Separation Oracle Model

We now apply our structural insights to the Distributed Separation Oracle (DSO) model. Recall that in this setting, servers hold implicit convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$ and do not exchange explicitly represented constraints. Instead, the network interacts purely via geometric queries: given a point x and a server index i, the server either certifies $x \in \mathcal{C}_i$ or returns a separating hyperplane. Hence, a naïve application of classic cutting-plane methods, e.g., Yudin & Nemirovskii (1976); Grötschel et al. (2012), requires finding a violated constraint at every iteration, which would require a prohibitive linear $\Omega(s)$ dependence per step. We show that one can break this global inspection barrier.

3.1 General DSO: Sublinear Queries via Stochastic Exploration

We first consider the general feasibility problem without bit-precision assumptions. In this continuous regime, exact optimization is not possible, so we instead simply seek a feasible point in $\mathcal{C} = \bigcap_{i=1}^{s} \mathcal{C}_i$ under a mild volume assumption $\mathrm{Vol}(\mathcal{C}) \geq \varepsilon$. To bypass querying all s servers, we establish two structural properties using the hidden low-dimensional geometry of the constraint system.

1. **Bounded Active Complexity:** Some $\mathcal{B} \subseteq \mathcal{S}$ with $|\mathcal{B}| \leq 2d^2$ satisfies $\mathrm{Vol}(\bigcap_{\mathcal{C}_i \in \mathcal{B}} \mathcal{C}_i) \leq C_d \,\mathrm{Vol}(\mathcal{C})$ for a dimension-dependent constant C_d (Lemma E.1).
2. **Guaranteed Volume Reduction:** Querying near the feasible region's center ensures that each returned separating half-space removes a constant fraction of its volume (Lemma E.2).

These properties reveal that at any stage, only a constant-sized, unknown subset of servers $\mathcal{B}$ is actually responsible for constraining the geometry. We formalize the challenge of identifying these servers as an online stochastic hidden-structure game. In this abstraction, there are s variables (servers), but only a hidden subset of $k \leq 2d^2$ are actually "active" (capable of returning a violation):

Problem 3.1. There are s light bulbs, k of which belong to an unknown subset B. The process runs for N rounds. In each round $t \in [N]$, an unknown subset S_t of bulbs is turned on, with $S_t \cap B \neq \emptyset$. At the start of each round, all bulbs are hidden. We may query bulbs adaptively within the round to determine whether they are on. After some number of queries, we must output a bulb $a_t \in S_t$. The goal is to succeed in all N rounds while minimizing the total number of queries across all rounds.

Because the active subset is hidden and dynamic, any deterministic sequence of queries is vulnerable to an adversarial oracle that hides the violation on the last queried server. We overcome this by utilizing a randomized exploration strategy: rather than maintaining a precise central point, the algorithm maintains the current feasible region W and samples a point $x \in$ uniformly at random.

Lemma 3.2. *There is an algorithm that solves the light bulb game in expected queries $O\left(\frac{k}{\alpha} s + k s^{\alpha} N \log s\right)$.*

Fixing the dimension d to be constant, then the basis corresponding to the set B in the light bulb game also has size $k = O(1)$. Moreover, since the volume of the feasible region is at least ε and the volume of the region bounded by the constraints decreases by a constant factor for each successful iteration, then we only require $N = O\left(\log \frac{1}{\varepsilon}\right)$ successful iterations. Hence, we have Theorem 1.3.

3.2 Bit-Limited DSO: Exact Optimization via Diophantine Discretization

We now turn to the BL-DSO model, where the input inequalities, objective, and oracle-returned separating hyperplanes have L-bit coefficients. We seek an exact minimizer of ϕ, using the active-server framework of Section 2. Every optimization subproblem retains the public box $Q = [-1, 1]^d$ and uses the same lexicographic tie-breaking rule.

To execute the meta-algorithm, we must solve the restricted subproblem (Step 2) over the dynamically sampled subset $\mathcal{T} \cup \mathcal{R}$ using only oracle queries. We achieve this by running the Center-of-Gravity (CoG) method on the sampled servers. However, CoG is a continuous geometric procedure that only provides approximate volume reduction. To achieve exact optimization, we introduce a combinatorial Diophantine discretization step.

Each non-empty subproblem is a bounded polytope defined by L-bit inequalities, so its lexicographically selected optimizer is a vertex of the subproblem's polytope belonging to $\Delta_{L,d}$. We maintain a convex localization region $W \subseteq Q$ containing this optimizer. Appendix I gives a positive lower bound $\tau = (d+1)^{-11Ld^4}$ on the volume of every non-degenerate k-dimensional simplex with vertices in $\Delta_{L,d}$. Once the Center-of-Gravity method reduces the volume of the current k-dimensional region W below τ, the remaining candidates $W \cap \Delta_{L,d}$ cannot contain $k+1$ affinely independent points, since their simplex would lie in W and have volume at least τ. We therefore replace W by $\mathrm{conv}(W \cap \Delta_{L,d})$ and continue within its affine hull, reducing the affine dimension. Incorporating this approach into the meta-algorithm from Section 2 gives our main result for the BL-DSO model:

Theorem 1.1. *Let $\mathcal{C}_1, \ldots, \mathcal{C}_s \subseteq \mathbb{R}^d$ be constraint sets in the BL-DSO model, with non-empty intersection $\mathcal{C} = \bigcap_{i=1}^s \mathcal{C}_i \subseteq [-1, 1]^d$. Let $\phi(x) = \langle c, x \rangle$ be a known linear functional with L-bit coefficients. There is a randomized algorithm that returns an exact minimizer $x_* \in \mathcal{C}$ of ϕ, using $O(sd) + L\sqrt{s} \cdot \mathrm{poly}(d)$ separation oracle queries in expectation.*

The main intuition is that the algorithm requires $O(d)$ rounds in expectation by Lemma 2.3, each focusing on a small active subset of $O(d\sqrt{s})$ servers. Within a round, the Diophantine-CoG routine shrinks the localization region and reduces its affine dimension, using $\mathrm{poly}(d) \cdot L$ steps in total, with each step querying only the active servers. After finding the subproblem's exact optimizer x_*, a pass over all s servers verifies global feasibility, incurring an additional $O(s)$ queries.

Furthermore, for constant dimension, we tighten the bound by introducing a second level of sampling within each active subproblem. We repeatedly sample $O_d(1)$ servers according to adaptively updated weights, optimize over the sample using Center-of-Gravity, and check the candidate against the active servers. This procedure takes $O_d(\log s$ iterations in expectation, each using $O_d(L + \sqrt{s})$ queries. Together with the outer global checks, this establishes the upper bound in Theorem 1.2. We defer the full procedure and analysis to Appendix J.

4 Communication-Efficient Distributed Optimization

We now shift from query complexity to classical communication complexity. In this setting, the structural representations are explicit: each of the s servers holds a convex set $\mathcal{C}_i$ given as the intersection of n_i half-spaces, with a total dataset size of $n = \sum_{i=1}^s n_i$. All constraints and the objective functional ϕ are specified to L bits of precision. The goal is to optimize ϕ exactly while minimizing the total number of bits communicated across the network.

Prior approaches, such as the state-of-the-art distributed adaptation of Clarkson's algorithm Clarkson (1995) by Vempala et al. (2020), repeatedly simulate global constraint aggregation. Because every iteration of their algorithm requires reweighting and broadcasting over the entire network, their communication bounds suffer from a multiplicative penalty coupling the network scale and the local dataset size, e.g., $\tilde{O}(sd \log n)$ in the blackboard model. We first describe how to use our framework from Section 2 to decouple the dataset scale $\log n$ from the network scale s in the total communication complexity.

Coordinator model. We first describe the implementation of our framework in the coordinator model. In each of the expected $O(d)$ outer iterations, the coordinator samples an active set $\mathcal{R}$ of $O(d\sqrt{s})$ servers and runs a subroutine (FINDOPT) to find the optimal solution x_* in the region defined by $\mathcal{T} \cup \mathcal{R}$, as well as the corresponding basis of at most d constraints. The coordinator broadcasts this $O(d^2L)$-bit basis to all s servers to check for global feasibility. This translates to Theorem 1.4.

In particular, the outer meta-algorithm executes $O(d)$ times. Within each round, running the FINDOPT subroutine on a subsystem of $O(d\sqrt{s})$ servers requires $L\sqrt{s} \cdot \mathrm{poly}(d \log n)$ communication. After finding the candidate x_*, explicitly sending the active basis to all s servers costs $s \cdot O(d^2L)$ bits. The claimed communication bound then follows from summing these contributions across the $O(d)$ outer rounds.

Blackboard model. In the blackboard model, communication is even cheaper because the information defining x_* is written on a shared blackboard, eliminating the need to explicitly route x_* to each server. However, we can push the communication efficiency significantly further by optimizing how we solve the subproblem on $\mathcal{T} \cup \mathcal{R}$.

Instead of calling the subroutine on all $O(d^2\sqrt{s})$ active servers simultaneously, we introduce a nested multiplicative-weights subroutine (NESTEDMWU, c.f., Appendix J). Inside each outer active-set round, NESTEDMWU maintains a probability distribution over the $O(d^2\sqrt{s})$ active servers, repeatedly sampling a multiset $\mathcal{B}$ of $6d^2$ servers independently with replacement according to this distribution. The subroutine then computes the lexicographically smallest optimum on $\mathcal{B}$ within $[-1, 1]^d$, and checks feasibility against the active set $\mathcal{T} \cup \mathcal{R}$, doubling the weights of violated active constraints only when their total weight is at most $1/(3d)$ of the total active weight. This inner-loop reweighting amplifies the probability of selecting violated basis constraints. By Lemma J.3, NESTEDMWU requires only $O(d \log sd)$ nested iterations in expectation to certify feasibility for the entire active subsystem. This gives Theorem 1.5. The communication analysis is given in Lemma J.5.

5 EMPIRICAL EVALUATIONS

In this section, we present a small-scale empirical study to complement our theoretical results. We emphasize that the purpose of these experiments is not to provide a fully optimized system evaluation, but rather to demonstrate the communication behavior of our analysis in a controlled setting.

ℓ_∞ regression on synthetic dataset. We evaluate distributed robust Chebyshev (ℓ_∞) regression, a standard formulation for worst-case regression under noise and outliers. In this setting, a network of s servers is given, where each server i holds a local dataset $X_i \in \mathbb{R}^{n_i \times d}$ with associated labels $y_i \in \mathbb{R}^{n_i}$. The objective is to compute a shared model $w \in \mathbb{R}^d$ that minimizes the maximum residual $\|X_i w - y_i\|_\infty$ across all data points. Introducing an auxiliary variable $t \in \mathbb{R}$ yields an equivalent linear program:

$$\min_{w,t} \; t \quad \text{such that} \quad \|X_i w - y_i\|_\infty \le t, \; \forall i \in [s].$$

Equivalently, by defining $z = [w^\top, t]^\top \in \mathbb{R}^{d+1}$, each server encodes its local constraints in standard affine form $A_i z \le b_i$. Each data point contributes two half-space constraints, so server i induces $2n_i$ linear inequalities. This defines a local convex feasibility set $\mathcal{C}_i$ per server, and the global solution is the intersection of these sets across all servers. For each server $i \in [s]$, we generate a local dataset consisting of feature-label pairs (X_i, y_i). The feature matrices are sampled as $X_i \sim \mathcal{N}(0, I)$, and labels are generated according to a noisy linear model $y_i = X_i w_{\text{true}} + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, 0.1^2)$. We fix the feature dimension to $d = 3$, so that the lifted LP dimension is $D = 4$, focusing on a low-dimensional regime that isolates communication effects rather than statistical performance. Each server holds n_i samples, and we vary both the local dataset size n_i and the number of servers s across experiments.

Experimental setup. All experiments are conducted in a centralized simulator that tracks only abstract communication costs, ignoring system-level effects such as stragglers or bandwidth constraints. We measure *Global Synchronization Rounds*, defined as the number of full-network barrier operations in which all s servers participate. We compare our proposed HYBRIDCLARKSON framework (Algorithms 8 and 9) against the distributed linear programming baseline of Vempala et al. (2020). For the baseline, we follow the standard distributed LP framework in which communication is driven by global sampling over all explicit constraints. For our method, each server is ac-

cessed through a separation-oracle-style interface that returns only a binary feasibility indicator, and the algorithm maintains a small dynamically selected active set of servers. All inner optimization problems are solved exactly using the HiGHS simplex solver via `scipy.optimize.linprog`, purely to avoid numerical artifacts. Our empirical evaluations were implemented in Python 3.11.5 and executed on a 64-bit system equipped with an Intel Core i7-3770 CPU at 3.40 GHz, 16GB RAM, and a 1TB SATA3 SSD. To verify statistical robustness and account for randomness in both the algorithms and the data generation process, every configuration is evaluated over 10 independent trials using freshly sampled datasets. We report the mean across these trials, with shaded regions representing ± 1 standard deviation.

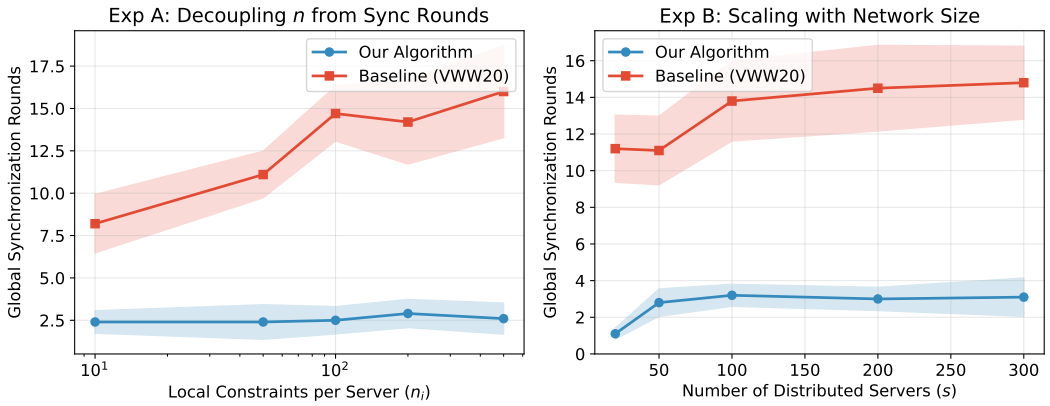

Figure 1: Global synchronization rounds in distributed Chebyshev regression ($D = 4$). Left: varying n_i with fixed $s = 50$. Right: varying s with fixed $n_i = 50$. Our method consistently requires fewer synchronization rounds than the baseline.

Results and discussion. We now discuss the observed communication behavior under different scaling regimes. We first fix the number of servers to $s = 50$ and vary the local dataset size n_i. In this regime, the baseline method Vempala et al. (2020) operates over the full set of explicit linear constraints, so its communication cost increases with the total number of constraints across all servers. In contrast, our method interacts with each server only through a separation oracle that provides a binary feasibility response. As a result, local dataset size does not directly influence the number of global synchronization rounds. This behavior is reflected in Figure 1 (Left), where the number of synchronization rounds remains essentially stable as n_i increases. Furthermore, the extremely narrow shaded regions for our algorithm demonstrate that this decoupling holds robustly across independent data distributions and sampling randomness, whereas the baseline exhibits much higher variance.

Next, we fix $n_i = 50$ and vary the number of servers s. The baseline exhibits increasing synchronization overhead and variance as the network grows, reflecting its dependence on global coordination over all servers. Our method instead maintains a small active subset of servers of size $\mathcal{O}(D\sqrt{s})$ during each iteration. This leads to fewer global synchronization events in practice, as shown in Figure 1 (Right). While the dependence on s is not completely eliminated empirically, the observed growth is noticeably milder than that of the baseline, and our active set framework drastically mitigates the high-variance stochastic fluctuations seen in the baseline's global sampling approach.

Summary. Overall, these experiments are intended as a simple validation of the communication patterns suggested by our theoretical analysis. They are not designed as a comprehensive empirical study or system-level benchmark. Nevertheless, the results are consistent with the predicted separation between local dataset size, network scale, and global synchronization complexity.

AI Use Statement Generative AI tools assisted with developing theoretical results, editing, and checking calculations. Algorithmic approaches, high-level proof strategies, literature review, and implementation details were conducted by the authors. The authors reviewed and validated all technical claims, proofs, experiments, and final content and take full responsibility for the work.

REFERENCES

Sepehr Assadi, Nikolai Karpov, and Qin Zhang. Distributed and streaming linear programming in low dimensions. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on principles of database systems*, pp. 236–253, 2019.

Leonard Mascot Blumenthal. Theory and applications of distance geometry. *(No Title)*, 1970.

Kenneth L Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995.

Daniel Dadush, Christopher Hojny, Sophie Huiberts, and Stefan Weltge. A simple method for convex optimization in the oracle model. *Mathematical Programming*, 206(1):283–304, 2024.

David Gale. Neighborly and cyclic polytopes, 1963.

Bernd Gärtner and Emo Welzl. Linear programming—randomization and abstract frameworks. In *Annual Symposium on Theoretical Aspects of Computer Science*, pp. 667–687. Springer, 1996.

Ambros Gleixner and Daniel E Steffy. Linear programming using limited-precision oracles. *Mathematical Programming*, 183(1):525–554, 2020.

Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

Branko Grünbaum. Partitions of mass-distributions and of convex bodies by hyperplanes., 1960.

Fritz John. Extremum problems with inequalities as subsidiary conditions. In *Traces and emergence of nonlinear programming*, pp. 197–215. Springer, 2013.

Jiří Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, New York, 2002.

R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1986.

Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

Santosh S Vempala, Ruosong Wang, and David P Woodruff. The communication complexity of optimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1733–1752. SIAM, 2020.

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.

A OUR MODELS

A.1 OPTIMIZATION

Linear programming. In the distributed linear programming problem that we consider, there are s closed convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s \subseteq \mathbb{R}^d$, which we will think of as being held on different servers. There is also a linear functional [2] ϕ, and our goal is output a point $x_* \in \bigcap_{i=1}^{s} \mathcal{C}_i$ that minimizes ϕ.

Typically in linear programming, one considers only linear constraints, as opposed to arbitrary convex constraints that we allow here. However, recall that one can express any convex set as an intersection of (possibly infinitely many) linear constraints. So our setting is effectively linear programming, but with no explicit limitations on the number of linear constraints that each server holds. While this is the most general model that we consider, some of our results will additionally assume a finite number of linear constraints held on each server.

[2] This means that we can express ϕ as $\phi(x) = \langle v, x \rangle$ for some $v \in \mathbb{R}^d$.

Convex feasibility. In the convex feasibility problem, the setup is the same as above, but there is no objective functional. Given closed convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$ whose intersection $\mathcal{C} = \bigcap_{i=1}^{s} \mathcal{C}_i$ is promised to be non-empty, the goal is to output a point $x_* \in \mathcal{C}$. We use "feasibility" in this search sense throughout. Our linear-optimization algorithms also solve this problem, under their stated assumptions, by taking $\phi \equiv 0$.

Bit considerations. Throughout, we will often assume that the convex sets $\mathcal{C}_i$ can be expressed as an intersection of half-spaces, each expressed with L bits of precision. Formally, we will say that a half-space

$$H = \{x : \langle x, v \rangle + b \geq 0\}$$

is expressed to L bits of precision if the entries of $v \in \mathbb{R}^d$ as well as $b \in \mathbb{R}$ lie in the set $\{\frac{i}{2^L} : i \in \mathbb{Z} \cap [-2^L, 2^L]\}$. We also say that a linear functional $\phi(x) = \langle x, v \rangle$ has L bits of precision if the entries of v are in this set. Note that this means that all coefficients lie in $[-1, 1]$. This is not a serious limitation though, as we can always rescale the sets $\mathcal{C}_i$. This is assumed only to simplify the statements of our results.

We make this bit precision assumption to make it simpler to quantify the communication between servers. Moreover, many of our results aim for an exact solution to the underlying linear program, and so we need some assumption on bit complexity in order to achieve this with finite communication. In many practical situations, each server could simply send their linear constraints to floating point precision if an approximate solution was desired.

A.2 Distributed models

We consider several related models for distributed linear programming, and introduce a model which appears to be new.

Distributed Separation Oracle Model. We introduce a new model for studying convex problems that are distributed among servers.

Definition A.1. In the Distributed Separation Oracle Model (DSO Model), there are s closed convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$ in $\mathbb{R}^d$. A query consists of picking an index i, along with a point $x \in \mathbb{R}^d$. If $x \in \mathcal{C}_i$, then the result of the query is $\perp$. Otherwise the result of the query is a separating hyperplane presented as a pair (v, b) with $v \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $\langle v, x \rangle + b > 0$ and $\langle v, y \rangle + b \leq 0$ for all $y \in \mathcal{C}_i$.

In the Bit-Limited DSO model (BL-DSO), we assume that each of the convex sets $\mathcal{C}_i$ is an intersection of half-spaces expressed with L bits of precision, and that each separating hyperplane returned by a query is expressed with L bits of precision.

While the separation oracle model is standard in optimization, to our knowledge this is the first time that a distributed version of this model has been described.

Coordinator Model. In the coordinator model, there is a central coordinator who can communicate bidirectionally with each of the s servers. Each message send from the coordinator to a server or vice versa counts towards the total communication.

This model is similar to the *point-to-point* model of communication where all pairs of servers are allowed to communicate with one another. Communication complexity in the point-to-point model is equivalent to the coordinator model up to a factor of 2, since in the coordinator model, we can simulate sending a message between two servers by routing through the coordinator.

Blackboard Model. In the blackboard model, the s servers are allowed to write on a central blackboard which is visible to all servers. Each bit written on the blackboard counts towards the total communication. This model is intended to model a communication setting where individual servers are able to broadcast a message to all other servers simultaneously.

B Additional Related Work

Other works have considered distributed optimization problems. In particular, Assadi et al. (2019) studies linear programming in the distributed and streaming models. However they have memory (and communication in the distributed setting) that is at least $s \log n$ for any number of rounds, while we achieve communication with the leading term in the large s regime independent of n. We achieve this in our model by being able to combine multiple linear constraints into a single convex constraint and generalize Clarkson's algorithm to such combined constraints. This is unique to the distributed model and not possible in the streaming model. Our approach yields an advantage when the servers hold convex constraints that are not naturally described by a small number of linear constraints. Along these lines, we also introduce and analyze the distributed separation oracle model. This model does not make assumptions about the number of linear constraints defining each convex set, and therefore requires our new techniques.

C Preliminaries

Lemma C.1 (Affine Farkas' Lemma)**.** *Schrijver (1986); Rockafellar (1970) Let $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^d$. Suppose the system $Ax \geq b$ is feasible. Then the following are equivalent:*

1. *$Ax \geq b$ implies $\langle c, x\rangle \geq \delta$ for every $x \in \mathbb{R}^d$.*
2. *There exists $y \in \mathbb{R}^m$ such that*

$$y \geq 0, \qquad A^\top y = c, \qquad \langle b, y\rangle \geq \delta.$$

Theorem C.2 (Helly's theorem)**.** *Matoušek (2002) Let $\mathcal{F}$ be a finite family of convex sets in $\mathbb{R}^d$. If the intersection of all sets in $\mathcal{F}$ is empty, then there exists a subfamily $\mathcal{S} \subseteq \mathcal{F}$ of size at most $d+1$ such that $\bigcap \mathcal{S} = \emptyset$.*

C.1 Clarkson-type procedure (algorithmic background)

For completeness, we describe a distributed variant of Clarkson's algorithm as used in prior work Vempala et al. (2020), which serves as a baseline primitive in parts of our analysis.

We first recall the standard blackboard implementation of Clarkson's algorithm for linear programming.

Algorithm 1 One round of Clarkson's Algorithm Vempala et al. (2020)

1. Sample $9d^2$ constraints uniformly from the multiset $\mathcal{H}$ held across all servers.
2. Solve the induced linear program and broadcast the solution x_*.
3. Each server reports the number of violated constraints it holds.
4. If no violations are reported, return x_*.
5. If the number of violated constraints exceeds $\frac{2}{9d+1}|\mathcal{H}|$, discard the round.
6. Otherwise, double the weight of all violated constraints in $\mathcal{H}$.

A standard argument shows that this procedure terminates in expectation after $O(d \log n)$ rounds, yielding an $O(sd \log n)$ communication bound in the blackboard model. The logarithmic dependence arises from repeated reweighting over the full constraint multiset.

A subtle but important issue is that the optimal solution may not be unique. To ensure deterministic behavior, we assume a fixed tie-breaking rule, e.g., selecting the lexicographically smallest optimal solution.

Role in our algorithms. In our work, this procedure is not used in its original global form. Instead, it appears only as a subroutine applied to *restricted subinstances involving only a small subset of servers*. This restriction is crucial: the effective number of participating servers in each invocation is reduced to $O(d\sqrt{s})$, which prevents the $\log n$ dependence from accumulating across all s servers.

Guarantee summary. The key property of Clarkson's method used in our analysis is the following: every successful round identifies at least one constraint from a basis of size at most d, and thus progress is certified after a bounded number of structural updates. This basis property ensures that only $O(d)$ successful rounds are required at the top level of our framework, independent of the total number of constraints n.

D MISSING PROOFS

Lemma 2.1. *Let $Q \subseteq \mathbb{R}^d$ be a compact convex set, and let $\mathcal{S}$ be a finite family of closed convex sets such that $Q \cap \bigcap_{\mathcal{C}\in\mathcal{S}} \mathcal{C} \neq \emptyset$. Let ϕ be a linear functional. For each $\mathcal{A} \subseteq \mathcal{S}$, let $x_{\mathcal{A}}$ be the lexicographically smallest minimizer of ϕ over*

$$P_{\mathcal{A}} := Q \cap \bigcap_{\mathcal{C}\in\mathcal{A}} \mathcal{C}, \qquad P_{\emptyset} := Q.$$

There exists $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ with $|\tilde{\mathcal{S}}| \leq d$ such that $x_{\tilde{\mathcal{S}}} = x_{\mathcal{S}}$.

Proof. Let $x_* := x_{\mathcal{S}}$, and order points by increasing objective value, breaking ties lexicographically by their coordinates. Let U be the set of points that precede x_* in this order.

The set U is convex. Indeed, for $u, v \in U$, consider the first comparison at which either point differs from x_*. At this comparison, both values are at most that of x_* and at least one is strictly smaller; all earlier comparisons are equal. Since the objective and coordinates are linear, every convex combination of u and v also precedes x_*.

By optimality, $U \cap P_{\mathcal{S}} = \emptyset$. Helly's theorem therefore gives a subfamily $\mathcal{B} \subseteq \{U, Q\} \cup \mathcal{S}$ of at most $d+1$ sets with empty intersection. Every set other than U contains x_*, so $U \in \mathcal{B}$. Set

$$\tilde{\mathcal{S}} := \mathcal{B} \setminus \{U, Q\}.$$

Then $\tilde{\mathcal{S}} \subseteq \mathcal{S}$, $|\tilde{\mathcal{S}}| \leq d$, and $U \cap P_{\tilde{\mathcal{S}}} = \emptyset$. Since $x_* \in P_{\tilde{\mathcal{S}}}$, it remains the selected optimizer, giving $x_{\tilde{\mathcal{S}}} = x_{\mathcal{S}}$. □

Lemma 2.2. *With the notation and assumptions of Lemma 2.1, fix $\mathcal{T} \subseteq \mathcal{S}$ and choose $\mathcal{R}$ uniformly from $\binom{\mathcal{S}}{r}$, where $1 \leq r \leq |\mathcal{S}|$. For $\mathcal{V}_{\mathcal{T},\mathcal{R}} = \{\mathcal{C} \in \mathcal{S} : x_{\mathcal{T}\cup\mathcal{R}} \notin \mathcal{C}\}$, we have $\mathbb{E}[|\mathcal{V}_{\mathcal{T},\mathcal{R}}|] \leq d|\mathcal{S}|/r$.*

Proof. For $R \in \binom{\mathcal{S}}{r}$ and $S \in \mathcal{S}$, let $\chi(S, R)$ be the indicator of $x_{\mathcal{T}\cup R} \notin S$. We have

$$\begin{aligned}
\binom{|\mathcal{S}|}{r} \mathbb{E}\,|\mathcal{V}_{\mathcal{T},R}| &= \sum_{R\in\binom{\mathcal{S}}{r}} \sum_{S\in\mathcal{S}} \chi(S,R) \\
&= \sum_{R\in\binom{\mathcal{S}}{r}} \sum_{S\in\mathcal{S}\setminus R} \chi(S,R) \\
&= \sum_{U\in\binom{\mathcal{S}}{r+1}} \sum_{S\in U} \chi(S, U\setminus\{S\}).
\end{aligned}$$

By Lemma 2.1, there is a basis $\mathcal{B} \subseteq \mathcal{T} \cup U$ of size at most d such that $x_{\mathcal{B}} = x_{\mathcal{T}\cup U}$. For $S \in U \setminus \mathcal{B}$, we have $\mathcal{B} \subseteq \mathcal{T} \cup (U \setminus \{S\}) \subseteq \mathcal{T} \cup U$. Thus the middle family has the same selected optimizer, which belongs to S. Consequently,

$$\sum_{S\in U} \chi(S, U\setminus\{S\}) \leq |\mathcal{B}| \leq d.$$

Plugging this in above and rearranging, we have

$$\mathbb{E}\,|\mathcal{V}_{\mathcal{T},R}| \leq \frac{\binom{|\mathcal{S}|}{r+1} d}{\binom{|\mathcal{S}|}{r}} = \frac{|\mathcal{S}| - r}{r+1} d \leq \frac{d\,|\mathcal{S}|}{r}.$$

□

Lemma 2.3. *The active-server meta-algorithm terminates almost surely and returns the lexicographically smallest minimizer x_* of ϕ over the full feasible region. It performs at most $d+1$ successful iterations and $O(d)$ total iterations in expectation.*

Proof. When the algorithm terminates, x_* is globally feasible and is the selected optimizer of a relaxation of the full problem, so it is the selected global optimizer. Thus the main task is to bound the number of iterations of the `repeat` block.

Say that an iteration is successful when $|\mathcal{V}| \le 2\sqrt{s}$.

Let $\mathcal{B}$ be the basis guaranteed by Lemma 2.1, so that $|\mathcal{B}| \le d$ and $x_{\mathcal{B}} = x_{\mathcal{S}}$. We claim that after N successful non-terminal iterations, $\mathcal{T}$ contains at least N members of $\mathcal{B}$. Indeed, the candidate x_* satisfies all constraints in $\mathcal{T}$. If it also satisfied $\mathcal{B}$, it could not precede $x_{\mathcal{B}} = x_{\mathcal{S}}$ in the fixed optimization order. Conversely, since $x_{\mathcal{S}}$ is feasible for the sampled subproblem, x_* could not follow $x_{\mathcal{S}}$. Thus $x_* = x_{\mathcal{S}}$, contradicting the fact that the procedure does not terminate. Hence x_* violates a member of $\mathcal{B} \setminus \mathcal{T}$, which is added to $\mathcal{T}$ if the iteration is successful. There are therefore at most d successful intermediate iterations, followed by one terminal iteration.

Next we analyze the probability of success. Conditioned on the preceding iterations, $\mathcal{T}$ is fixed and $\mathcal{R}$ is a fresh uniform sample of size $r = \min\{s, \lceil d\sqrt{s} \rceil\}$. If $r = s$, then $\mathcal{V} = \emptyset$. Otherwise, Lemma 2.2 bounds the conditional expected number of violations by

$$\frac{ds}{|\mathcal{R}|} = \frac{ds}{\lceil d\sqrt{s} \rceil} \le \sqrt{s}.$$

By Markov's inequality, each iteration is successful with conditional probability at least $\frac{1}{2}$.

The probability of t consecutive failures is consequently at most 2^{-t}, so the conditional expected waiting time for the next successful iteration is at most $\sum_{t \ge 0} 2^{-t} = 2$. Since at most $d+1$ successful iterations are needed, the expected total number of iterations is at most $2(d+1) = O(d)$, and termination occurs with high probability. □

E Separation Oracle Queries

E.1 Convex Set Lemmas

Lemma E.1. *Let $\mathcal{C}_1, \dots, \mathcal{C}_s$ be closed convex sets in $\mathbb{R}^d$, where $d \ge 1$, with compact intersection $\mathcal{C} := \bigcap_{i=1}^s \mathcal{C}_i$ of positive volume. There is a subfamily $\mathcal{S}$ of these sets of size at most $2d^2$ such that*

$$\operatorname{Vol}\left(\bigcap \mathcal{S}\right) \le C_d \operatorname{Vol}(\mathcal{C}),$$

where C_d depends only on d.

Proof. We first bound $\mathcal{C}$ by a polytope with volume not too much larger. By John's ellipsoid theorem John (2013), there is an ellipsoid $E_0 \subseteq \mathcal{C}$, with center z, such that

$$\mathcal{C} \subseteq E := z + d(E_0 - z).$$

Consequently,

$$\operatorname{Vol}(E) = d^d \operatorname{Vol}(E_0) \le d^d \operatorname{Vol}(\mathcal{C}).$$

Let P be the bounding box of E in its principal-axis coordinates. Writing κ_d for the volume of the Euclidean unit ball in $\mathbb{R}^d$, we have

$$\operatorname{Vol}(P) = \frac{2^d}{\kappa_d} \operatorname{Vol}(E) \le \frac{(2d)^d}{\kappa_d} \operatorname{Vol}(\mathcal{C}).$$

Let $H_1, \dots, H_{2d}$ be the closed half-spaces defining P. We claim that for each j, there is a subfamily $\mathcal{S}_j \subseteq \{\mathcal{C}_1, \dots, \mathcal{C}_s\}$ of size at most d whose intersection is contained in H_j.

To see this, let $U_j = \mathbb{R}^d \setminus H_j$, which is an open half-space. Since $\mathcal{C} \subseteq H_j$, the finite family $\{\mathcal{C}_1, \dots, \mathcal{C}_s, U_j\}$ has empty intersection. By Helly's theorem, Theorem C.2, some subfamily of at

most $d+1$ sets has empty intersection. Every subfamily consisting only of the $\mathcal{C}_i$'s contains $\mathcal{C} \neq \emptyset$, so this witness must include U_j. Its remaining sets form a subfamily $\mathcal{S}_j$ of size at most d satisfying

$$U_j \cap \bigcap \mathcal{S}_j = \emptyset, \qquad \text{hence} \qquad \bigcap \mathcal{S}_j \subseteq H_j.$$

Taking $\mathcal{S} = \bigcup_{j=1}^{2d} \mathcal{S}_j$ gives $|\mathcal{S}| \leq 2d^2$ and

$$\bigcap \mathcal{S} \subseteq \bigcap_{j=1}^{2d} H_j = P.$$

The volume bound follows from $C_d = (2d)^d/\kappa_d$. □

Lemma E.2. *Let $\mathcal{C}$ be a compact convex set in $\mathbb{R}^d$ with nonempty interior. There is a measurable subset $\mathcal{R} \subseteq \mathcal{C}$ with $\mathrm{Vol}(\mathcal{R}) \geq c_d \mathrm{Vol}(\mathcal{C})$ such that, for every half-space $\mathcal{H}$ with $\partial\mathcal{H} \cap \mathcal{R} \neq \emptyset$, we have*

$$\mathrm{Vol}(\mathcal{H} \cap \mathcal{C}) \geq c_d \mathrm{Vol}(\mathcal{C}).$$

Here $c_d > 0$ depends only on d.

Proof. By John's ellipsoid theorem John (2013), the maximum-volume inscribed ellipsoid E of $\mathcal{C}$ satisfies

$$\mathrm{Vol}(E) \geq d^{-d} \mathrm{Vol}(\mathcal{C}).$$

Let z be its center and set $\mathcal{R} = z + \frac{1}{2}(E - z)$. Then $\mathcal{R} \subseteq \mathcal{C}$ and

$$\mathrm{Vol}(\mathcal{R}) = 2^{-d} \mathrm{Vol}(E) \geq (2d)^{-d} \mathrm{Vol}(\mathcal{C}).$$

To bound the volume cut off by a half-space, apply an invertible affine transformation taking E to $B(0,1)$ and $\mathcal{R}$ to $B(0,1/2)$. Write the transformed half-space as $\mathcal{H} = \{x : \langle u, x\rangle \geq t\}$, where $\|u\|_2 = 1$. Since its boundary meets $B(0,1/2)$, we have $|t| \leq 1/2$. For every $y \in B(3u/4, 1/4)$,

$$\|y\|_2 \leq 1, \qquad \langle u, y\rangle \geq \frac{1}{2} \geq t.$$

Thus $\mathcal{H} \cap B(0,1)$ contains a ball of radius $1/4$ and has volume at least $4^{-d}\mathrm{Vol}(B(0,1))$. Since invertible affine transformations preserve volume ratios, in the original coordinates we obtain

$$\mathrm{Vol}(\mathcal{H} \cap \mathcal{C}) \geq \mathrm{Vol}(\mathcal{H} \cap E) \geq 4^{-d} \mathrm{Vol}(E) \geq (4d)^{-d} \mathrm{Vol}(\mathcal{C}).$$

Both conclusions therefore hold with $c_d = (4d)^{-d}$. □

E.2 Separation Oracle Query Algorithm

We first recall the following toy problem that will be critical to our separation oracle algorithm.

Problem 3.1. There are s light bulbs, k of which belong to an unknown subset B. The process runs for N rounds. In each round $t \in [N]$, an unknown subset S_t of bulbs is turned on, with $S_t \cap B \neq \emptyset$. At the start of each round, all bulbs are hidden. We may query bulbs adaptively within the round to determine whether they are on. After some number of queries, we must output a bulb $a_t \in S_t$. The goal is to succeed in all N rounds while minimizing the total number of queries across all rounds.

We emphasize that the query set q_t may itself be constructed through multiple adaptive interactions with the light bulbs in round t, e.g., q_t can first query a set of light bulbs in round t and then based on those responses, query additional light bulbs in round t.

We give a simple algorithm for solving this toy problem, and then show that it gives rise to an algorithm for the distributed separation oracle model in low dimensions.

Lemma 3.2. *There is an algorithm that solves the light bulb game in expected queries $O\left(\frac{k}{\alpha}s + ks^{\alpha}N\log s\right)$.*

Proof. We assume $s \geq 2$, since for $s = 1$ the only bulb is always on and can be output without any queries. We maintain a set $\mathcal{S}$ of candidate hidden configurations, initially $\mathcal{S} = [s]^k$, and we always preserve the invariant that a tuple listing the true hidden set remains in $\mathcal{S}$. Whenever we discover that a tuple $(x_1, \ldots, x_k) \in \mathcal{S}$ is entirely off, we remove it from $\mathcal{S}$.

We set $m = \lceil s^\alpha \ln s \rceil$ for a parameter $\alpha > 0$. At the start of each round, let $\mathcal{S}_0 = \mathcal{S}$. We sample $\min\{m, |\mathcal{S}_0|\}$ elements uniformly from $\mathcal{S}_0$ without replacement and query the k bulbs associated with each sample, spending at most km queries. We cache responses within each round, so each bulb is queried at most once. If any queried bulb is on, we output it and immediately move to the next round. Otherwise, we fall back to querying all remaining unqueried bulbs, output an on bulb, and retain precisely the tuples in $\mathcal{S}_0$ containing at least one bulb revealed to be on.

Call a round *bad* if none of the sampled configurations contains an on bulb, while at least a $\frac{1}{s^\alpha}$ fraction of $\mathcal{S}_0$ contains a bulb that is on. Otherwise, we call the round *good*. If all configurations are sampled, a bad round is impossible. Otherwise, conditioned on the preceding history and the current on/off configuration, sampling without replacement gives that the probability a round is bad is at most

$$(1 - s^{-\alpha})^m \leq e^{-m/s^\alpha} \leq \frac{1}{s}.$$

Each such round costs at most s queries, so the total expected cost from all bad rounds is $O(N)$.

Now consider good rounds. A good round is either cheap, meaning it costs at most km queries, or it is informative in the sense that the fallback query leaves fewer than $|\mathcal{S}_0|/s^\alpha$ candidate configurations. Thus an informative round decreases the size of $\mathcal{S}$ by at least a factor of s^α relative to its size at the start of the round. Since $\mathcal{S}$ starts with size s^k, never increases, and always contains a true hidden configuration, this second type of event can only happen $O\left(\frac{k}{\alpha}\right)$ times. Each such decrease costs $O(s)$ queries, contributing $O\left(\frac{k}{\alpha}s\right)$ overall. The remaining good rounds contribute $O(kmN) = O(ks^\alpha N \log s)$. Putting everything together, for any fixed $\alpha > 0$ the expected query complexity is

$$O\left(\frac{k}{\alpha}s + ks^\alpha N \log s\right),$$

as claimed. □

Theorem E.3. *Let $\mathcal{C}_1, \ldots, \mathcal{C}_s$ be closed convex sets contained in the unit ball of $\mathbb{R}^d$, each accessible via separation oracle queries. Let $\mathcal{C} = \bigcap_i \mathcal{C}_i$ and suppose that $\mathrm{Vol}(\mathcal{C}) \geq \varepsilon > 0$. For every $\alpha > 0$, there is an algorithm that outputs a point $x_* \in \mathcal{C}$ with probability at least $3/4$ and makes at most*

$$O\left(\frac{s}{\alpha} + s^\alpha \log(s+1) \log\left(2 + \frac{1}{\varepsilon}\right)\right)$$

separation oracle queries. The implicit constant depends only on d and α.

Proof. Fix the dimension d to be constant. Our algorithm maintains a region $\mathcal{W}$, initially the unit ball, satisfying the separating half-space constraints observed so far. By Lemma E.1, there is a subset B of $k = \min\{s, 2d^2\}$ servers, enlarging the subset if necessary, such that

$$\mathcal{K} := \bigcap_{j \in B} \mathcal{C}_j, \qquad \mathrm{Vol}(\mathcal{K}) \leq C_d \,\mathrm{Vol}(\mathcal{C}).$$

On each iteration i, sample x_i uniformly from $\mathcal{W}$ and use the light-bulb procedure to find a violated server. If all servers accept x_i, return it. For $s = 1$, simply query the sole server.

For analysis only, let T be the first iteration with $x_T \in \mathcal{K}$. Before T, every iteration violates a server in B, so the light-bulb invariant holds. The algorithm does not need to know B or T.

We first bound T. Let Z_i be the volume of $\mathcal{W}$ after iteration i if $T > i$, and zero otherwise. By Lemma E.2, conditional on the preceding history, the query point belongs to the central region with probability at least c_d. In that event, either $T = i$ or the returned cut removes at least a c_d fraction of the volume: apply the lemma to the excluded half-space translated through x_i. Thus, writing $\rho = 1 - c_d^2$ and $\kappa_d = \mathrm{Vol}(B(0,1))$, we have

$$\mathbb{E}[Z_i] \leq \rho^i \kappa_d.$$

Since $Z_N \geq \varepsilon$ whenever $T > N$, choosing

$$N = \left\lceil \frac{\ln(2\kappa_d/\varepsilon)}{-\ln \rho} \right\rceil = O_d\left(\log\left(2 + \frac{1}{\varepsilon}\right)\right)$$

gives $\Pr[T > N] \leq \frac{1}{2}$.

Conditioned on the history before a round and on $x_i \in \mathcal{K}$, its probability of being feasible is

$$\frac{\mathrm{Vol}(\mathcal{C})}{\mathrm{Vol}(\mathcal{W} \cap \mathcal{K})} \geq \frac{1}{C_d}.$$

Averaging over the possible first hitting iterations therefore gives

$$\Pr[T \leq N,\ x_T \in \mathcal{C}] \geq \frac{1}{2C_d}.$$

Finally, we analyze the query bound. Writing $m = \lceil s^\alpha \ln(s+1) \rceil$, the proof of Lemma 3.2 bounds exploration before T by kmN queries, informative fallback rounds by ks/α queries, and the expected cost of bad rounds by N. Iteration T costs at most s additional queries. Hence the expected query count through $\min\{T, N\}$ is at most

$$M := kmN + \frac{ks}{\alpha} + N + s = O_d\left(\frac{s}{\alpha} + s^\alpha N \log(s+1)\right).$$

For $s = 1$, the same bound holds because at most N queries are made.

We shall abort a trial after N iterations or before exceeding $\lceil 4C_d M \rceil$ queries. By Markov's inequality, the probability that the analyzed prefix exceeds this budget is at most $\frac{1}{4C_d}$. Each trial therefore succeeds with probability at least $\frac{1}{4C_d}$. Restarting from the initial state with fresh randomness for $O(C_d)$ trials reduces the failure probability to at most $\frac{1}{4}$, since the same success bound holds conditional on preceding trials. The total query count is deterministically $O_d(M)$, giving the claimed bound. □

F Our Variant of Clarkson's Algorithm

We next give algorithms in both the coordinator model and the blackboard that improve upon the algorithms of Vempala et al. (2020) when the number of servers s is large.

For now we black-box the `FindOPT` procedure and give a general analysis of the communication complexity.

F.1 Coordinator model

Theorem F.1. *Let N_{comm} be an upper bound on the communication complexity of each call to `FindOPT`. Then Algorithm 2 uses $O(sd^3L + dN_{comm})$ communication in the coordinator model.*

Proof. By the previous result, the repeat block runs $O(d)$ times in expectation. Moreover sending x_* to a given server can be achieved with $O(d^2L)$ communication. We simply send the d linear constraints which for which the optimum is x_*. Each linear constraint takes $O(dL)$ communication, so $O(d^2L)$ communication overall.

Each round therefore requires at most $O(sd^2L + N_{\text{comm}})$ communication. There are $O(d)$ rounds, and so we use $O(sd^3L + dN_{\text{comm}})$ communication overall. □

F.2 Blackboard model

We give a similar algorithm in the blackboard model of communication, the main difference being that communicating x_* is now easier. The reason is that the `FindOPT` procedure will write the constraints defining x_* on the blackboard. Thus we do not need the extra communication of sending x_* to the servers, as they have enough information to compute x_* from the blackboard. This algorithm is otherwise a direct adaptation of the coordinator model algorithm. See the appendix for the pseudocode.

Algorithm 2 DistributedClarksonCoordinator

input s servers holding convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$, linear functional ϕ
  $\mathcal{T} \leftarrow \emptyset$
  repeat
    $\mathcal{R} \leftarrow$ uniformly random sample of $\lceil d\sqrt{s} \rceil$ server indices
    $x_* \leftarrow \texttt{FindOPT}(\mathcal{T} \cup \mathcal{R}, \phi)$
    $\mathcal{V} \leftarrow \emptyset$
    for $i = 1, \ldots, s$ **do**
      Send (the constraints defining) x_* to server i
      if $x_* \notin \mathcal{C}_i$ **then**
        Insert i into $\mathcal{V}$
      end if
    end for
    if $|\mathcal{V}| \leq 2\sqrt{s}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{V}$
    end if
  until $\mathcal{V} = \emptyset$
output x_*

Theorem F.2. *Algorithm 3 can be implemented to run with*

$$O((s + N_{comm})d)$$

communication in the blackboard model, where N_{comm} is the communication used by the procedure `FindOPT` *in the blackboard model, when run on a set of at most $O(d\sqrt{s})$ servers.*

Proof. Each round, involves a single bit of communication from each server, as well an application of the `FindOPT` procedure. By Theorem 2.3 the outer block runs only $O(d)$ times, and so the communication bound follows. □

G Exact solutions for Few Servers

G.1 Clarkson's Algorithm

We recall from Vempala et al. (2020) that a version of Clarkson's algorithm can be modified to run in the distributed setting. This version of Clarkson has worse than an sd^2 dependence in its leading term, so we do not apply it directly to solve our linear programs. Instead we will apply it as a subroutine on smaller subsets of the servers. The algorithm referred to below is given in Algorithm 1 above.

Theorem G.1. *(Vempala et al. (2020)) Suppose that each of s servers hold a convex set $\mathcal{C}_i$, each of which is expressed as an intersection of n_i half-spaces with bit precision L. Also suppose that there is a global linear functional ϕ expressible with the same bit precision. Let $n = \sum_i n_i$.*

There is an algorithm that uses

$$O((sd^2(L + \log d) + d^3 L + s \log n)d \log n)$$

communication in the coordinator model and

$$O((d^2(L + \log d) + d^3 L + s \log n)d \log n)$$

communication in the blackboard model, and outputs x_* optimizing ϕ.

In the coordinator model, the coordinator also receives a set of linear constraints such that the optimum of ϕ subject to those constraints defines x_*. In the blackboard model, these constraints are written on the blackboard.

Furthermore, these algorithms run in polynomial time.

Proof. See Theorem 10.1 of Vempala et al. (2020). □

Corollary G.2. *By using the algorithm of Theorem G.1 to implement* `FindOPT`*, Algorithm 2 uses* $O(sd^3L) + \mathrm{poly}(d \log n) \cdot L\sqrt{s}$ *communication in the coordinator model.*

Proof. When running `FindOPT` within Algorithm 2, the total number of servers involved in the procedure is at most $O(d\sqrt{s})$. Therefore, the total communication of this subroutine works out to

$$N_{\text{comm}} = O(d^4\sqrt{s}L \log n + d^2\sqrt{s}\log^2 n)$$

in the coordinator model.

By Theorem G.1, the total communication used by Algorithm 2 is at most

$$O(sd^3L + d^5\sqrt{s}L \log n + d^3\sqrt{s}\log^2 n).$$

□

We next describe the corresponding implementation in the blackboard model, given in Algorithm 3. In each round, `FindOPT` writes at most d input inequalities determining the candidate optimum on the shared blackboard. All subproblems retain the public box $Q = [-1, 1]^d$ and use the same lexicographic tie-breaking rule, so every server can reconstruct the candidate from these inequalities. Each server then contributes one bit to the global feasibility check. This basic implementation provides the starting point for the recursive refinement in the following subsection.

G.2 A RECURSIVE IMPROVEMENT

In the blackboard model, we observe that it is possible to tighten the lower order terms for small d. In particular, we can decouple the $\log(n)$ dependence from the $\sqrt{s}$, which may be valuable when d is small and n is extremely large. We present this version of the algorithm in Algorithm 4.

Algorithm 3 DistributedClarksonBlackboard

input s servers holding convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$, linear functional ϕ
  $\mathcal{T} \leftarrow \emptyset$
  repeat
    $\mathcal{R} \leftarrow$ uniformly random sample of $\lceil d\sqrt{s} \rceil$ server indices
    $x_* \leftarrow$ `FindOPT`$(\mathcal{T} \cup \mathcal{R}, \phi)$ ▷Retain Q and write a basis determining the lexicographically smallest optimum on the blackboard.
    $\mathcal{V} \leftarrow \emptyset$
    for $i = 1, \ldots, s$ **do**
      Server i reconstructs x_* by lexicographically minimizing $(\phi(x), x_1, \ldots, x_d)$ over Q subject to the returned basis inequalities
      Server i broadcasts the boolean $(x_* \notin \mathcal{C}_i)$
      if Server i sent `True` **then**
        Add i to $\mathcal{V}$
      end if
    end for
    if $|\mathcal{V}| \leq 2\sqrt{s}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{V}$
    end if
  until $\mathcal{V} = \emptyset$
output x_*

Theorem G.3. *Assuming an optimal solution lies in* $[-1, 1]^d$*, there is an algorithm that uses expected total communication*

$$O(sd) + (cd)^{O(1+\log\log(s+2))} L \log(n+1)$$

in the blackboard model, where $c > 1$ *is an absolute constant. In particular, for constant* d*, it uses*

$$O_d\left(s + (\log(s+2))^{p_d} L \log(n+1)\right)$$

expected communication, where $p_d > 0$ *depends only on* d*.*

Algorithm 4 RecursiveClarksonBlackboard

input s servers holding convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$, linear functional ϕ
  $\mathcal{T} \leftarrow \emptyset$
  if $s \leq 64d^2$ **then**
    return `FindOPT`$(\{\mathcal{C}_1, \ldots, \mathcal{C}_s\}, \phi)$
  end if
  repeat
    $\mathcal{R} \leftarrow$ uniformly random sample of $\lceil d\sqrt{s} \rceil$ server indices
    if $s \geq 2d$ **then**
      $x_* \leftarrow$ `RecursiveClarksonBlackboard`$(\mathcal{T} \cup \mathcal{R}, \phi)$
    else
      $x_* \leftarrow$ `FindOPT`$(\mathcal{T} \cup \mathcal{R}, \phi)$
    end if
    $\mathcal{V} \leftarrow \emptyset$
    for $i = 1, \ldots, s$ **do**
      Server i uses the constraints written on the blackboard by `FindOPT` to compute x_*
      if $x_* \notin \mathcal{C}_i$ **then**
        Insert i into $\mathcal{V}$
      end if
    end for
    if $|\mathcal{V}| \leq 2\sqrt{s}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{V}$
    end if
  until $\mathcal{V} = \emptyset$
output x_*

Proof. Let $T(s)$ be a non-decreasing upper bound on the expected communication used by Algorithm 4, with the original parameters n, L, d fixed. All calls retain $[-1, 1]^d$ and use the same lexicographic tie-breaking rule.

Recall that the outer loop runs $O(d)$ times in expectation. Sending x_* to the servers does not require additional communication in the blackboard model. The reason is that all relevant information is written on the blackboard, and so each server can compute x_* for themselves.

Since there are at most d successful non-terminal updates,

$$|\mathcal{T} \cup \mathcal{R}| \leq 2d\sqrt{s} + \lceil d\sqrt{s} \rceil \leq 4d\sqrt{s}.$$

For $s > 64d^2$, this is less than $s/2$, ensuring that recursion decreases the instance size. Announcing the sample and the violation set takes $O(s)$ bits. Thus, for an absolute constant $c \geq 2$,

$$T(s) \leq cd\, T(4d\sqrt{s}) + O(sd) \qquad (s > 64d^2).$$

For $s \leq 64d^2$, remove duplicate inequalities locally before calling `FindOPT`. The remaining number $n_0 \leq n$ of input inequalities is at most $64d^2 2^{O(dL)}$, so $\log n_0 = O(dL + \log d)$. Theorem G.1, with the lexicographic implementation from Lemma J.5, therefore gives the recurrence

$$T(s) \leq F := \mathrm{poly}(d) L \log(n + 1).$$

It remains to solve this recurrence. To simplify the notation, define $Q(s) = T(16d^2 s)$, so that

$$Q(s) \leq cd\, Q(\sqrt{s}) + O(sd^3),$$

with base case $s \leq 4$. For $s > 4$, let h be the smallest integer such that $s^{2^{-h}} \leq 4$. Iteratively expanding the recurrence gives

$$Q(s) \leq O(d^3) \sum_{j=0}^{h-1} (cd)^j s^{2^{-j}} + (cd)^h F.$$

Consecutive summands have ratio

$$\frac{cd}{s^{2^{-(j+1)}}},$$

which is at most $\frac{1}{2}$ whenever $s^{2^{-j}} \geq 4(cd)^2$. The initial terms sum to at most $2s$, while the remaining terms sum to at most

$$4(cd)^2 \sum_{j=0}^{h-1} (cd)^j \leq \frac{4(cd)^2}{cd-1}(cd)^h.$$

Consequently,

$$Q(s) \leq O(d^3 s) + (cd)^h \operatorname{poly}(d) L \log(n+1).$$

Since $T(s) = Q(s/(16d^2))$ and $h \leq \log_2 \log(s+2) + O(1)$, the desired claim follows. □

Theorem 1.6. *Suppose each of s servers holds a convex set $\mathcal{C}_i$, which is an intersection of n_i half spaces expressed to L bits of precision, and wish to optimize a linear functional ϕ expressed to L bits of precision. The communication guarantees of Theorem 1.4 and Theorem G.3 hold under the addition of a global convex constraint $\mathcal{G}$. That is, a server outputs x_* with $\phi(x_*)$ minimal in $\mathcal{G} \cap (\bigcap_i \mathcal{C}_i)$.*

Proof. Set $K = G \cap [-1,1]^d$, and retain K in every subproblem. The optimizer-containment assumption ensures that every subproblem has a nonempty compact feasible region. For a family $\mathcal{A}$ of input inequalities, let $x_{\mathcal{A}}$ be its lexicographically smallest optimizer over K.

By Lemma 2.1, every finite family $\mathcal{A}$ has a subfamily $\mathcal{B}$ of at most d inequalities with $x_{\mathcal{B}} = x_{\mathcal{A}}$. For an explicitly known sample of at most $9d^2$ inequalities, such a basis can be found locally by enumerating subfamilies of size at most d and testing whether their selected optimizer satisfies the entire sample. Existence follows from the basis lemma, and any passing subfamily preserves the selected optimizer. Communicating $\mathcal{B}$ costs $O(d^2 L)$ bits. Every recipient reconstructs the same point by optimizing locally over K subject to $\mathcal{B}$, regardless of the precision of its coordinates.

To implement `FindOPT`, apply the weighted Clarkson routine to the individual input inequalities, retaining K. The analysis of Lemma J.3 applies to this family: there are $O(d \log(n+1))$ iterations in expectation and at most $O(d \log(n+1))$ successful nonterminal weight updates. Consequently, all total weights and violated weights have $O(d \log(n+1))$-bit representations. Each server reports its aggregate weights, allowing the coordinator to sample globally by first selecting a server proportionally to its total weight and then sampling locally. Only $O(d^2)$ sampled inequalities are transmitted per iteration.

On m participating servers, one iteration therefore costs

$$O(d^3 L + m d^2 L + m d \log(n+1))$$

bits in the coordinator model and

$$O(d^3 L + m d \log(n+1))$$

bits in the blackboard model. These bounds include communicating the sampled inequalities, the candidate's defining basis, and the aggregate weights.

For Theorem 1.4, substituting $m = O(d\sqrt{s})$ gives the same subproblem communication bound. Each outer global check still costs $O(sd^2 L)$ bits, since the candidate is specified by at most d input inequalities.

For Theorem G.3, each recursive base case has $m = \operatorname{poly}(d)$ servers. The same local duplicate removal used in its proof gives base-case communication $\operatorname{poly}(d) L \log(n+1)$. All parties reconstruct recursive candidates from the blackboard, so the recurrence and its solution remain unchanged. Thus both claimed communication guarantees hold. □

G.3 Application to distributed ℓ_p regression

Finally, we describe applications of our approach to distributed ℓ_p regression.

Theorem 1.7. *Fix a constant $p > 2$ and let $0 < \varepsilon < 1/2$. Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ with L-bit entries distributed by rows among s servers, there is an algorithm that with constant probability finds $\hat{x}$ satisfying*

$$\|A\hat{x} - b\|_p \leq (1+\varepsilon) \cdot \min_x \|Ax - b\|_p .$$

The algorithm uses expected communication $\tilde{O}\left(s(d/\varepsilon^2)^3 L + L\sqrt{s}\cdot \mathrm{poly}(d/\varepsilon)\right)$ in the coordinator model and $\tilde{O}\left(sd/\varepsilon^2 + (d/\varepsilon^2)^{O(1+\log\log(s+2))}L\right)$ in the blackboard model. Here $\tilde{O}$ suppresses factors polylogarithmic in n, d, and $1/\varepsilon$, and hidden constants may depend on p.

Proof. Fix a constant $p > 2$, and write the rows of A as $a_1^\top, \dots, a_n^\top$. Let $\delta = \varepsilon/10$, and choose

$$R = \left\lceil C_p \frac{d\log(d/\varepsilon)}{\varepsilon^2} \right\rceil,$$

where C_p is a sufficiently large constant. Let x_{opt} minimize $\|Ax - b\|_p$. Such a minimizer exists: the affine residual space $\{Ax - b : x \in \mathbb{R}^d\}$ is closed, and its intersection with any bounded norm sublevel set is compact.

For each $j \in [R]$ and $i \in [n]$, independently sample E_{ji} from the exponential distribution of mean one, and set $w_{ji} = E_{ji}^{-1/p}$. Define

$$F(x) = \sum_{j=1}^{R} \max_{i\in[n]} w_{ji}|a_i^\top x - b_i|.$$

By (Vempala et al., 2020, Theorem 8.2), with probability at least $q_p > 0$, both

$$F(x_{\mathrm{opt}}) \le (1+\delta)\gamma_p R\|Ax_{\mathrm{opt}} - b\|_p$$

and

$$F(x) \ge (1-\delta)\gamma_p R\|Ax - b\|_p \qquad \text{for every } x \in \mathbb{R}^d$$

hold, where $\gamma_p = \mathbb{E}[E_{11}^{-1/p}]$ and q_p depend only on p.

We next give the weights bounded precision. Choose an integer $C \ge 4/q_p$, set $B = CnR$, and let $h = \lceil \log_2(B/\delta)\rceil$. Replace each weight by

$$\widehat{w}_{ji} = 2^{-h}\left\lfloor 2^h \min\{B, \max\{B^{-1}, w_{ji}\}\}\right\rfloor.$$

These weights are positive, since $2^{-h} \le \delta/B < B^{-1}$. Moreover,

$$\Pr\left[w_{ji} \notin [B^{-1}, B]\right] \le e^{-B^p} + B^{-p} \le \frac{2}{B}.$$

A union bound therefore shows that clipping changes no weight with probability at least $1 - q_p/2$. On this event, rounding introduces error at most $2^{-h} \le \delta/B \le \delta w_{ji}$, and hence

$$(1-\delta)w_{ji} \le \widehat{w}_{ji} \le w_{ji}.$$

Writing

$$\widehat{F}(x) = \sum_{j=1}^{R} \max_{i\in[n]} \widehat{w}_{ji}|a_i^\top x - b_i|,$$

we consequently have

$$(1-\delta)F(x) \le \widehat{F}(x) \le F(x) \qquad \text{for every } x.$$

The objective $\widehat{F}$ attains its minimum. Indeed, if $w_{\min} = \min_{j,i}\widehat{w}_{ji} > 0$, then

$$\widehat{F}(x) \ge Rw_{\min}\|Ax - b\|_\infty.$$

Thus its sublevel sets in the closed affine residual space are compact. Let $\widehat{x}$ be an exact minimizer. With probability at least $q_p/2$, the reduction's guarantees and the event that clipping changes no weight hold simultaneously. On this event,

$$\begin{aligned}
\|A\widehat{x} - b\|_p &\le \frac{F(\widehat{x})}{(1-\delta)\gamma_p R} \\
&\le \frac{\widehat{F}(\widehat{x})}{(1-\delta)^2\gamma_p R} \\
&\le \frac{\widehat{F}(x_{\mathrm{opt}})}{(1-\delta)^2\gamma_p R} \\
&\le \frac{1+\delta}{(1-\delta)^2}\|Ax_{\mathrm{opt}} - b\|_p \\
&\le (1+\varepsilon)\|Ax_{\mathrm{opt}} - b\|_p.
\end{aligned}$$

For the last inequality, substituting $\delta = \varepsilon/10$ gives

$$(1+\varepsilon)(1-\varepsilon/10)^2 - (1+\varepsilon/10) = \frac{\varepsilon(70 - 19\varepsilon + \varepsilon^2)}{100} > 0.$$

We compute $\widehat{x}$ through the linear program

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{R} t_j \\ \text{subject to} \quad & \widehat{w}_{ji}(a_i^\top x - b_i) \le t_j, \\ & -\widehat{w}_{ji}(a_i^\top x - b_i) \le t_j \qquad (i \in [n],\ j \in [R]). \end{aligned}$$

This program has $D = d + R$ variables and $N = 2nR$ inequalities. Each server generates the weights and inequalities associated with its own input rows locally. The program's optimum is attained, because for each x its minimum over the auxiliary variables is exactly $\widehat{F}(x)$.

Each rounded weight has a dyadic representation using $O(\log(nR/\varepsilon))$ bits. Under our dyadic input-precision convention, multiplying these weights by the input coefficients and scaling the inequalities by a public power of two gives coefficient precision at most a publicly known integer

$$\ell = O_p\left(L + \log(nD/\varepsilon)\right).$$

We choose this scaling so that all coefficients have magnitude at most one and are integer multiples of $2^{-\ell}$.

We now establish a public box containing an optimal solution. Write the D variables as $y = u - v$, where $u, v \ge 0$. The resulting polyhedron in $2D$ variables has an optimal vertex. To see this, first minimize the sum of its nonnegative coordinates over its nonempty optimal face. This minimum is attained, since imposing an upper bound on that sum produces a compact set. The minimum-sum face is compact; successively minimizing its coordinates selects a singleton face, which is a vertex of the original split polyhedron.

At this vertex, the active constraint normals span $\mathbb{R}^{2D}$. Otherwise, a nonzero direction orthogonal to every active normal would permit sufficiently small feasible perturbations in both directions, contradicting that the point is a vertex. Select $2D$ linearly independent active inequalities and clear their dyadic denominators. All entries of the resulting integer system have magnitude at most 2^ℓ. Cramer's rule therefore bounds every coordinate of the vertex by

$$U = (2D)!\, 2^{2D\ell},$$

because its denominator is a nonzero integer and each numerator is a determinant with at most $(2D)!$ terms, each of magnitude at most $2^{2D\ell}$. Since $u, v \ge 0$, the corresponding original variables satisfy $|u_i - v_i| \le U$.

Retain the public box $[-U, U]^D$ in every optimization subproblem. It contains a global optimizer, so every subproblem is nonempty and compact. Apply the public-domain implementation established above, using the same lexicographic tie-breaking rule throughout. Candidates are communicated through their defining input inequalities, each of precision at most ℓ; the public box requires no communication.

Set $\Lambda = L + \log(nD/\varepsilon)$. Applying Theorem 1.4 and Theorem G.3, with the public-domain implementation, gives expected communication

$$O(sD^3\Lambda) + \Lambda\sqrt{s} \cdot \mathrm{poly}(D \log(N+1))$$

in the coordinator model and

$$O(sD) + (cD)^{O(1+\log\log(s+2))}\Lambda \log(N+1)$$

in the blackboard model. These bounds hold over all realizations of the rounded weights.

Finally,

$$D = O_p\left(\frac{d\log(d/\varepsilon)}{\varepsilon^2}\right), \qquad N = 2nR, \qquad \Lambda = \tilde{O}_p(L),$$

which gives the stated coordinator bound. For the blackboard bound, we further note that $D \le (d/\varepsilon^2)^{O_p(1)}$, to obtain the stated exponent. The unsuppressed term $O(sD)$ is independent of n, as claimed. □

H THE SEPARATION ORACLE MODEL

In this section, we consider the BL-DSO model, where the input inequalities, objective, and oracle-returned separating hyperplanes have L-bit coefficients. We assume that the feasible region is non-empty and contained within the public box $Q = [-1, 1]^d$, which is retained in every optimization subproblem.

In order for our Clarkson variant to run correctly, we need to find the lexicographically smallest optimal solution x_* each round for the active subset of $O(d\sqrt{s})$ servers. We will accomplish this with a cutting-plane method. Initially, this only reduces the region containing x_* to one of very small intrinsic volume. Diophantine considerations then imply that the remaining candidate vertices in $\Delta_{L,d}$ lie in a lower-dimensional affine subspace. Restricting to their convex hull and repeating within its affine hull allows us to find x_* exactly. We discuss these Diophantine considerations in Appendix I.

H.1 THE CENTER-OF-GRAVITY ALGORITHM

We recall the well-known center-of-gravity algorithm. We give a version here which is suited to our communication setting for optimizing a linear functional ϕ.

The key lemma which allows the center-of-gravity algorithm to work is as follows.

Proposition H.1. *Grünbaum (1960) Let V be a convex region in $\mathbb{R}^d$ with center of mass z. Let H be half-space with z on its boundary. Then $\mathrm{Vol}(H \cap V)$ is at most $\left(1 - \frac{1}{e}\right) \mathrm{Vol}(V)$.*

In Algorithm 5 and Algorithm 6, we use x_* to denote the lexicographically smallest minimizer of ϕ over $Q \cap \bigcap_{j=1}^{s} \mathcal{C}_j$, where $Q = [-1, 1]^d$. Centers of gravity are taken with respect to intrinsic volume.

Algorithm 5 CenterOfGravity

input s convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$ accessible via separation oracle queries, linear functional ϕ, compact convex region R containing x_*, of positive affine dimension, number of iterations N
  $R_0 \leftarrow R$, $k \leftarrow \dim(\mathrm{aff}(R))$
  for $i = 1, 2, \ldots, N$ **do**
    if $\dim(\mathrm{aff}(R_{i-1})) < k$ **then**
      return R_{i-1}
    end if
    Let z_{i-1} be the center of gravity of R_{i-1}
    Perform a separation oracle query on z_{i-1} to all convex sets
    if some separation oracle returned a half-space H **then**
      $R_i \leftarrow R_{i-1} \cap H$
    else
      Let g be the first functional in $(\phi, x \mapsto x_1, \ldots, x \mapsto x_d)$ that is nonconstant on $\mathrm{aff}(R_{i-1})$
      $R_i \leftarrow R_{i-1} \cap \{x : g(x) \leq g(z_{i-1})\}$
    end if
  end for
output R_N

The candidate set $V \cap \Delta_{L,d}$, its convex hull, and its affine hull can be computed by enumerating the finite public set $\Delta_{L,d}$ and using the explicit description of V. These operations require no additional oracle queries.

Lemma H.2. *Suppose that ϕ is a linear functional corresponding to an inner product against a vector in $\Sigma_{L,d}$, and that each $\mathcal{C}_1, \ldots, \mathcal{C}_s$ is an intersection of closed half-spaces whose boundary hyperplanes belong to $\mathcal{H}_{L,d}$. Let $Q = [-1, 1]^d$ and assume that $P = Q \cap \bigcap_{j=1}^{s} \mathcal{C}_j$ is non-empty.*

Algorithm 6 finds the lexicographically smallest minimizer x_* of ϕ over P in the BL-DSO model using $\mathrm{poly}(d) \cdot (sL)$ queries.

Proof. First, P is a non-empty compact polytope, since $\mathcal{H}_{L,d}$ is finite and $P \subseteq Q$. Successively minimizing $\phi, x_1, \ldots, x_d$ over P defines a unique point x_*. This point is a vertex: if $x_* = (u+v)/2$

Algorithm 6 FindOptCOG

input s BL-DSO constraint sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$ accessible via separation oracle queries, linear functional ϕ, $Q \cap \bigcap_{j=1}^{s} \mathcal{C}_j \neq \emptyset$
  $V \leftarrow Q = [-1, 1]^d$
  Set $N \leftarrow 1 + \left\lceil \frac{\log((2\sqrt{d})^d / \tau_{L,d})}{-\log(1-1/e)} \right\rceil$
  while $\dim(\mathrm{aff}(V)) > 0$ **do**
    $V \leftarrow \texttt{CenterOfGravity}(\mathcal{C}_1, \ldots, \mathcal{C}_s, \phi, V, N)$
    $V \leftarrow \mathrm{conv}(V \cap \Delta_{L,d})$
  end while
output the unique element of V

for $u, v \in P$, successive minimality forces $\phi(u) = \phi(v) = \phi(x_*)$ and then $u_j = v_j = x_{*,j}$ for every coordinate j.

The active constraint normals at x_* span $\mathbb{R}^d$. Otherwise, choose a nonzero vector orthogonal to all active normals. Since the finitely many inactive inequalities have positive slack, sufficiently small perturbations of x_* in both directions along this vector remain in P, contradicting extremality. Thus x_* is the unique intersection of d linearly independent active hyperplanes. The facets of Q also belong to $\mathcal{H}_{L,d}$, so $x_* \in \Delta_{L,d}$.

We next show that every maintained region contains x_*. This holds initially because $x_* \in Q$. Every oracle-returned half-space contains x_*. For an objective cut, let z be the feasible centroid of the current region. If ϕ is non-constant on the current affine hull, set $g = \phi$. Otherwise, let j be the smallest index for which the coordinate function $x \mapsto x_j$ is non-constant on that affine hull, and set $g(x) = x_j$. In the latter case, ϕ and the coordinates $x_1, \ldots, x_{j-1}$ are constant on the current affine hull, so their values agree at z and x_*. Since $z \in P$, optimality and the lexicographic tie-breaking rule imply $g(x_*) \leq g(z)$ in either case. Thus the objective cut preserves x_*. Finally, replacing a region R by $\mathrm{conv}(R \cap \Delta_{L,d})$ preserves x_* because $x_* \in R \cap \Delta_{L,d}$. All maintained regions also remain contained in Q.

Consider a call to `CenterOfGravity` starting in affine dimension $k \geq 1$. If the dimension drops, the returned region already has dimension less than k. Otherwise, each iteration reduces its k-dimensional volume by a factor of at most $\eta := 1 - 1/e$. Indeed, an oracle cut is contained in the parallel half-space through the centroid, and an objective cut passes through the centroid and is nonconstant on the current affine hull. Grünbaum's bound therefore applies within that affine hull.

Since $Q \subseteq B(0, \sqrt{d})$, every k-dimensional section of Q lies in a k-dimensional cube of side $2\sqrt{d}$. Consequently, after N iterations without a dimension drop, the returned region R satisfies

$$\mathrm{Vol}_k(R) \leq \eta^N (2\sqrt{d})^k \leq \eta^N (2\sqrt{d})^d < \tau_{L,d},$$

by the choice of N in Algorithm 6. If $R \cap \Delta_{L,d}$ contained $k+1$ affinely independent points, their convex hull would be a k-dimensional simplex contained in R. Proposition I.2 would give this simplex volume at least $\tau_{L,d}$, a contradiction. Thus $\mathrm{conv}(R \cap \Delta_{L,d})$ has dimension less than k.

In either case, each iteration of Algorithm 6 decreases the affine dimension while preserving x_*. After at most d iterations, the maintained region is therefore the singleton $\{x_*\}$.

The candidate set and its convex and affine hulls can be computed from the finite public set $\Delta_{L,d}$ and the explicit maintained region, without additional oracle queries. Each call to `CenterOfGravity` uses at most sN queries, where $N = O(Ld^4 \log d)$. The total query count is therefore $O(sLd^5 \log d) = \mathrm{poly}(d) \cdot (sL)$. □

H.2 Clarkson's algorithm via separation oracles

Theorem H.3. *Let $\mathcal{C}_1, \ldots, \mathcal{C}_s$ be BL-DSO constraint sets whose intersection is non-empty and contained in $Q = [-1, 1]^d$, and let ϕ be a known linear functional with L-bit coefficients. Algorithm 7 returns the lexicographically smallest minimizer of ϕ over this intersection using $O(sd + \sqrt{s}\,\mathrm{poly}(d)L)$ separation oracle queries in expectation.*

Algorithm 7 SeparationOracleClarkson

input s servers holding convex sets $\mathcal{C}_1, \ldots, \mathcal{C}_s$, linear functional ϕ
  $\mathcal{T} \leftarrow \emptyset$
  repeat
    $\mathcal{R} \leftarrow$ uniformly random sample of $\lceil d\sqrt{s} \rceil$ server indices
    $x_* \leftarrow \texttt{FindOptCOG}(\mathcal{T} \cup \mathcal{R}, \phi)$
    $\mathcal{V} \leftarrow \emptyset$
    for $i = 1, \ldots, s$ **do**
      Perform a separation oracle query (x_*, i)
      if $x_* \notin \mathcal{C}_i$ **then**
        Insert i into $\mathcal{V}$
      end if
    end for
    if $|\mathcal{V}| \leq 2\sqrt{s}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{V}$
    end if
  until $\mathcal{V} = \emptyset$
output x_*

Proof. Correctness and the expected $O(d)$ iterations of the outer `repeat` block follow from Lemma 2.3. There are at most d successful non-terminal iterations, each adding at most $2\sqrt{s}$ servers to $\mathcal{T}$. Thus each iteration calls `FindOptCOG` on $O(d\sqrt{s})$ servers and then performs $O(s)$ additional separation oracle queries. By Lemma H.2, each such call uses $\mathrm{poly}(d) \cdot L\sqrt{s}$ queries. Multiplying by the expected number of outer iterations gives the stated bound. □

Similar to Theorem G.3 we can also give a recursive improvement, which gives tight bounds for constant d.

For constant dimension, we improve the bound by using the weighted-sampling routine `NestedMWU` from Appendix J to solve each active subproblem.

Theorem H.4. *For constant d, let $\mathcal{C}_1, \ldots, \mathcal{C}_s$ be BL-DSO constraint sets whose intersection is non-empty and contained in $Q = [-1, 1]^d$. For a known linear functional ϕ with L-bit coefficients, there is an algorithm that returns its lexicographically smallest minimizer over this intersection using $O_d(s + L \log s)$ separation oracle queries in expectation.*

Proof. Replace each call to `FindOptCOG` in Algorithm 7 by `NestedMWU`, using `FindOptCOG` as its exact optimization subroutine. Every optimization call retains Q and uses the same lexicographic tie-breaking rule.

On a working set of m servers, the weighted-sampling routine performs $O_d(\log m)$ iterations in expectation, as analyzed in Appendix J. Each iteration optimizes over a sample of $O_d(1)$ servers and checks the resulting point against all m servers. By Lemma H.2, its query cost is therefore $O_d(L + m)$.

By Lemma 2.3, the outer algorithm performs $O_d(1)$ iterations in expectation, and every working set has size $m = O_d(\sqrt{s})$. Including the global feasibility checks, the expected query complexity is

$$O_d\left(s + (L + \sqrt{s}) \log s\right) = O_d\left(s + L \log s\right).$$

□

This bound is essentially optimal for constant d, as even finding a feasible point requires $\Omega(s + L)$ queries. We prove this for algorithms that succeed with probability at least $3/4$ on every non-empty instance.

For the $\Omega(s)$ bound, it suffices to consider the input space $\mathbb{R}$. Choose a uniformly random server $J \in [s]$ and an independent uniform bit $b \in \{0, 1\}$. Let $\mathcal{C}_J = \{b\}$ and $\mathcal{C}_i = [0, 1]$ for $i \neq J$. Until server J is queried, the responses reveal no information about b, so the probability of returning the unique feasible point is at most $\frac{1}{2}$. Moreover, the probability of querying J within the first t queries

is at most $\frac{t}{s}$: before that query, the execution follows the same path as on the instance with every set equal to $[0, 1]$. Writing T for the number of queries, we obtain

$$\Pr(\text{correct}) \le \frac{1}{2} + \frac{t}{s} + \Pr(T > t).$$

Taking $t = \lfloor s/8 \rfloor$ shows that $\mathbb{E}[T] = \Omega(s)$.

For the $\Omega(L)$ bound, let $M = 2^L$ and consider one server holding $\mathcal{C}_1 = \{b\}$, where $b \in \{0, 1/M, \dots, (M-1)/M\}$. For a query $z \in [0, 1]$, the oracle reports feasibility if $z = b$. Otherwise, it returns

$$\begin{cases} \{x : x \ge (\lfloor Mz \rfloor + 1)/M\}, & z < b, \\ \{x : x \le (\lceil Mz \rceil - 1)/M\}, & z > b. \end{cases}$$

These are valid separating half-spaces with L-bit coefficients. Queries outside $[0, 1]$ receive the fixed separating half-space $x \ge 0$ or $x \le 1$, respectively. Thus each query has at most three possible responses, determined by the comparison between z and b.

Choose b uniformly from its M possible values. For any fixed internal randomness, the resulting ternary decision tree has at most 3^t terminating transcripts using at most t queries, each of which can correctly identify at most one value of b. Consequently, also averaging over internal randomness,

$$\Pr(\text{correct}) \le \frac{3^t}{M} + \Pr(T > t).$$

Taking $t = \lfloor \frac{1}{2} \log_3 M \rfloor$ gives $\mathbb{E}[T] = \Omega(\log M) = \Omega(L)$. Adding servers holding $[0, 1]$ gives the same bound for any number of servers. Combining the two bounds yields $\Omega(\max\{s, L\}) = \Omega(s+L)$ worst-case expected queries, even for finding a feasible point.

I Diophantine Approximation Lemmas

The Clarkson variant in the previous section relied on the servers having access to the convex body as a set of explicit linear constraints. This does not work directly in the BL-DSO model. Here we obtain our BL-DSO query bounds by combining our Clarkson variant with the center-of-gravity algorithm to reduce the intrinsic volume of a localization region containing the selected optimal vertex. By making this region sufficiently small, we can apply Diophantine approximation ideas to show that the remaining candidate vertices lie in a lower-dimensional affine subspace. Restricting the region to their convex hull and repeating within its affine hull eventually isolates the selected optimal vertex.

For integers $L, d \ge 1$, let $\Sigma_{L,d}$ be the set of vectors in $\mathbb{R}^d$ whose entries lie in $\{\frac{a}{2^L} : a \in \mathbb{Z} \cap [-2^L, 2^L]\}$. Let $\mathcal{H}_{L,d}$ denote the set of hyperplanes in $\mathbb{R}^d$ that can be expressed as $x \in \mathbb{R}^d$ with $\langle x, v \rangle = b$, where $v \in \Sigma_{L,d} \setminus \{0\}$ and $b \in \{\frac{a}{2^L} : a \in \mathbb{Z} \cap [-2^L, 2^L]\}$. Let $\Delta_{L,d}$ be the set of points in $\mathbb{R}^d$ that can be expressed as the unique intersection of d hyperplanes in $\mathcal{H}_{L,d}$.

Proposition I.1. *Every element of $\Delta_{L,d}$ can be written in the form $\frac{p}{q}$, where $p \in \mathbb{Z}^d$ and $q \in \mathbb{Z}$ satisfy $\|p\|_\infty \le (d!)2^{Ld}$ and $1 \le q \le (d!)2^{Ld}$.*

Proof. Every $x \in \Delta_{L,d}$ can be expressed as $x = A^{-1}b$, where $A \in \mathbb{R}^{d \times d}$ is invertible and all entries of A and b lie in $\{\frac{a}{2^L} : a \in \mathbb{Z} \cap [-2^L, 2^L]\}$. By Cramer's rule, each coordinate satisfies $x_i = P_i/Q$, where $Q = \det A$ and P_i is the determinant obtained by replacing the ith column of A with b. Each such determinant is a degree-d polynomial in its entries with $d!$ terms and coefficients ± 1. Thus $P_i = \frac{R_i}{2^{Ld}}$ and $Q = \frac{R_0}{2^{Ld}}$, where $R_0, R_1, \dots, R_d$ are integers of absolute value at most $(d!)2^{Ld}$. Since A is invertible, $R_0 \ne 0$. Setting $q = |R_0|$ and $p = \operatorname{sgn}(R_0)(R_1, \dots, R_d)$ gives $x = p/q$ with the claimed bounds. □

Proposition I.2. *For $1 \le k \le d$, a non-degenerate k-dimensional simplex with vertices in $\Delta_{L,d}$ has k-dimensional volume at least $\tau := (d+1)^{-11Ld^4}$.*

Proof. Let v be the volume of such a simplex, with vertices $x_0, \dots, x_k$. By Proposition I.1, write $x_i = \frac{p_i}{q_i}$, where $p_i \in \mathbb{Z}^d$ and $1 \le q_i \le D := (d!)2^{Ld}$. By the Cayley-Menger formula (see e.g. Blumenthal (1970)), the squared volume is

$$v^2 = \frac{|\det M|}{(k!)^2 2^k},$$

where M is a $(k+2) \times (k+2)$ matrix whose first row and column are $(0, 1, \dots, 1)$ and whose remaining entries are the squared distances $\|x_i - x_j\|_2^2$.

Let $E = \operatorname{diag}(1, q_0^2, \dots, q_k^2)$ and $M' = EME$. Then M' has integer entries: its first row and column have entries $0, q_0^2, \dots, q_k^2$, and its remaining entries are

$$q_i^2 q_j^2 \|x_i - x_j\|_2^2 = \|q_j p_i - q_i p_j\|_2^2 \in \mathbb{Z}.$$

Since the simplex is non-degenerate, $\det M' \neq 0$. Thus $|\det M'| \ge 1$, and

$$|\det M| = \frac{|\det M'|}{\prod_{i=0}^k q_i^4} \ge D^{-4(k+1)}.$$

It follows that

$$v \ge \frac{1}{2^{k/2} k! D^{2(k+1)}}.$$

Using $d! \le d^d$, $k \le d$, and $L \ge 1$, we have

$$D \le (d+1)^{2Ld}, \qquad 2^{k/2} k! \le (d+1)^{2d}.$$

Consequently,

$$v \ge (d+1)^{-2d - 4Ld(k+1)} \ge (d+1)^{-10Ld^2} \ge (d+1)^{-11Ld^4},$$

where the second inequality uses $k + 1 \le 2d$ and $2d \le 2Ld^2$. □

J Optimized Distributed Protocol

In this section, we present an optimized distributed protocol, with better dependencies between s and $\log n$ factors. The algorithm is a distributed adaptation of Clarkson's sampling framework, designed to minimize coordination across the s servers. We give the algorithm in full in Algorithm 8. Throughout this appendix, we retain the fixed domain Q in every optimization problem, and assume that $Q \cap \bigcap_{\mathcal{C} \in \mathcal{S}} \mathcal{C}$ is non-empty. For $\mathcal{A} \subseteq \mathcal{S}$, let $x_{\mathcal{A}}$ denote the lexicographically smallest minimizer of ϕ over $Q \cap \bigcap_{\mathcal{C} \in \mathcal{A}} \mathcal{C}$. Every call to `FindOPT` returns this selected optimizer. For the BL-DSO application, take $Q = [-1, 1]^d$ and implement `FindOPT` using `FindOptCOG`.

At a high level, the outer loop of Algorithm 8 repeatedly solves small random subproblems and uses violations as feedback to refine future samples. In each iteration, the algorithm draws a random subset $\mathcal{R}$ of size $\tilde{O}(d\sqrt{s})$ and augments it with a set $\mathcal{T}$ of previously observed violations. The algorithm then computes a candidate solution on $\mathcal{R} \cup \mathcal{T}$ and checks feasibility against all servers. By standard sampling arguments, a solution that is feasible for a sufficiently large random subset is unlikely to violate many overall constraints. In other words, the total number of servers with violated constraints must be small. These violating constraints are added to $\mathcal{T}$, ensuring that they are carried forward into future iterations. Since the global optimum is defined by a basis of size at most d, this process quickly captures all critical constraints, after which the candidate solution becomes globally feasible and the algorithm terminates.

The inner routine, NestedMWU, described in Algorithm 9, reduces the cost of solving each sampled subproblem. Rather than invoking the optimization oracle on the entire working set $\mathcal{S}' = \mathcal{R} \cup \mathcal{T}$, the algorithm maintains weights over $\mathcal{S}'$ and repeatedly samples a small subset $\mathcal{B}$ of size $O(d^2)$ proportional to these weights. The oracle is applied only to $\mathcal{B}$ to produce a candidate solution, whose feasibility is then tested over all of $\mathcal{S}'$. If the total weight of violated constraints is too large, the sample is discarded. Otherwise, the weights of violated constraints are increased multiplicatively. This reweighting quickly amplifies the weights of constraints that are consistently violated, causing them to be included in future samples with high probability, similar to standard multiplicative

weights update arguments. In particular, constraints belonging to a basis for $\mathcal{S}'$ quickly dominate the sampling distribution, ensuring that the oracle is eventually applied to a representative subset. As a result, the inner loop finds a solution feasible for all of $\mathcal{S}'$ while only invoking the expensive optimization routine on a small number of samples.

Algorithm 8 HybridClarkson

input s servers $\mathcal{S} = \{\mathcal{C}_1, \ldots, \mathcal{C}_s\}$; dimension d
output $x^* \in \mathbb{R}^d$
  $\mathcal{T} \leftarrow \emptyset$
  repeat
    $\mathcal{R} \leftarrow$ uniform sample of $\lceil d\sqrt{s} \rceil$ servers
    $x^* \leftarrow \texttt{NestedMWU}(\mathcal{R} \cup \mathcal{T})$
    $\mathcal{V} \leftarrow \{\mathcal{C}_i \in \mathcal{S} : x^* \notin \mathcal{C}_i\}$
    if $|\mathcal{V}| \leq 2d\sqrt{s}$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{V}$
    end if
  until $\mathcal{V} = \emptyset$
output x^*

Algorithm 9 NESTEDMWU

input working set $\mathcal{S}'$
output x feasible for all $\mathcal{C}_i \in \mathcal{S}'$
  $w_i \leftarrow 1$ for all $\mathcal{C}_i \in \mathcal{S}'$
  repeat
    $W \leftarrow \sum_{\mathcal{C}_i \in \mathcal{S}'} w_i$
    $\mathcal{B} \leftarrow$ a multiset of $r = 6d^2$ independent samples from $\mathcal{S}'$, each selecting $\mathcal{C}_i$ with probability w_i/W
    $x \leftarrow \texttt{FindOPT}(\mathcal{B})$
    $\mathcal{V} \leftarrow \{\mathcal{C}_i \in \mathcal{S}' : x \notin \mathcal{C}_i\}$
    $W_{\mathcal{V}} \leftarrow \sum_{\mathcal{C}_i \in \mathcal{V}} w_i$
    if $W_{\mathcal{V}} \leq \frac{1}{3d} W$ **then**
      for each $\mathcal{C}_i \in \mathcal{V}$ **do**
        $w_i \leftarrow 2w_i$
      end for
    end if
  until $\mathcal{V} = \emptyset$
output x

Lemma J.1 (Deterministic Basis Violation)**.** *Let $\mathcal{A}$ be a finite set of convex constraints with optimal basis $\mathcal{B}^*_{\mathcal{A}} \subseteq \mathcal{A}$. Let x be the optimum of an arbitrary subproblem $\mathcal{B} \subseteq \mathcal{A}$. If x is infeasible for $\mathcal{A}$, i.e., it strictly violates at least one constraint in $\mathcal{A}$, then x must violate at least one constraint in the basis $\mathcal{B}^*_{\mathcal{A}}$.*

Proof. Suppose that x_B satisfies every constraint in $B^*_{\mathcal{A}}$. Since $x_{B^*_{\mathcal{A}}} = x_{\mathcal{A}}$, the point x_B cannot precede $x_{\mathcal{A}}$ in the order defined by objective value followed by lexicographic tie-breaking. Conversely, $x_{\mathcal{A}}$ is feasible for the problem on B, so x_B cannot follow $x_{\mathcal{A}}$ in this order. Thus $x_B = x_{\mathcal{A}}$, contradicting infeasibility for $\mathcal{A}$. □

Lemma J.2 (Exactness and Feasibility)**.** *Upon termination, the* HYBRIDCLARKSON *algorithm returns the lexicographically smallest minimizer x^* of ϕ over $Q \cap \bigcap_{i=1}^{s} \mathcal{C}_i$, with no violated constraints.*

Proof. We first show that x^* is feasible. The algorithm terminates when the global violation set is empty, i.e., $\mathcal{V}_{global} = \emptyset$. By construction, this set contains all constraints in $\mathcal{S}$ violated by x^*. Moreover, every optimization call retains Q. Thus $x^* \in Q \cap \bigcap_{\mathcal{C} \in \mathcal{S}} \mathcal{C}$.

It remains to show that x^* is the selected optimizer. The point x^* is produced by NESTEDMWU on the working set $\mathcal{S}' = \mathcal{R} \cup \mathcal{T}$. Let $\mathcal{B} \subseteq \mathcal{S}'$ be the set of constraints appearing in its final sample.

The call to FINDOPT returns the lexicographically smallest minimizer over $Q \cap \bigcap_{\mathcal{C}\in\mathcal{B}} \mathcal{C}$. At inner termination, x^* also satisfies every constraint in $\mathcal{S}'$. Since every point feasible for $\mathcal{S}'$ is feasible for $\mathcal{B}$, no such point can have a smaller objective value, or an equal objective value and lexicographically smaller coordinates. Therefore, x^* is the selected optimizer for $\mathcal{S}'$.

Finally, $\mathcal{S}' \subseteq \mathcal{S}$, so every globally feasible point is feasible for $\mathcal{S}'$. Since x^* is globally feasible, the same argument shows that it is the lexicographically smallest global minimizer. □

Lemma J.3 (Phase 2 Iteration Bound)**.** *For any non-empty working set $\mathcal{S}' \subseteq \mathcal{S}$, the* NESTEDMWU *subroutine in Algorithm 9 terminates almost surely and uses $\mathcal{O}(d \log(|\mathcal{S}'| + 1))$ iterations in expectation.*

Proof. Let $m = |\mathcal{S}'|$, and let $\mathcal{B}^*_{\mathcal{S}'}$ denote a basis preserving its selected optimizer, with $|\mathcal{B}^*_{\mathcal{S}'}| \le d$.

Condition on the preceding iterations, so that the weights w_i and their sum W are fixed. Each iteration samples a multiset $\mathcal{B}$ of size $r = 6d^2$ independently with replacement, selecting $\mathcal{C}_i$ with probability w_i/W. To bound the expected violated weight, consider $r + 1$ independent samples $C_1, \dots, C_{r+1}$ from this distribution. Let I_j indicate that the selected optimizer on all samples except position j violates C_j.

By Lemma 2.1, the full sample has a basis of at most d constraints. Choose one occurrence of each basis constraint. Deleting any other position retains the basis and therefore preserves the selected optimizer of the full sample, which satisfies the deleted constraint. Consequently,

$$\sum_{j=1}^{r+1} I_j \le d.$$

By exchangeability, all I_j have the same expectation. Conditioned on the first r samples, the probability that the independent last sample is violated is $W_{\mathcal{V}}/W$. Thus, with expectations conditional on the preceding iterations,

$$\frac{\mathbb{E}[W_{\mathcal{V}}]}{W} = \mathbb{E}[I_{r+1}] = \frac{1}{r+1}\mathbb{E}\left[\sum_{j=1}^{r+1} I_j\right] \le \frac{d}{r+1} \le \frac{1}{6d}.$$

We call an iteration *successful* if $W_{\mathcal{V}} \le W/(3d)$. By Markov's inequality, conditioned on any preceding history,

$$\Pr\left[W_{\mathcal{V}} > \frac{1}{3d}W\right] \le \frac{1/(6d)}{1/(3d)} = \frac{1}{2}.$$

Thus each iteration succeeds with conditional probability at least $\frac{1}{2}$.

In a successful iteration, the total weight increases by $W_{\mathcal{V}}$; unsuccessful iterations leave the weights unchanged. Starting from unit weights, after k successful non-terminal iterations we therefore have

$$W^{(k)} \le m\left(1 + \frac{1}{3d}\right)^k \le me^{k/(3d)}.$$

In every nonterminal iteration, the candidate violates some constraint of $\mathcal{S}'$. By Lemma J.1, it must therefore violate at least one constraint of $\mathcal{B}^*_{\mathcal{S}'}$. Consequently, every successful nonterminal iteration doubles the weight of at least one basis constraint. For $k > 0$, averaging over the at most d basis constraints shows that some basis constraint has been doubled at least k/d times. Its weight is at least $2^{k/d}$ and cannot exceed the total weight, so

$$2^{k/d} \le W^{(k)} \le me^{k/(3d)}.$$

Taking logarithms and rearranging gives

$$\frac{k}{d}(\ln 2 - 1/3) \le \ln m.$$

Since $\ln 2 - 1/3 > 0$, the number of successful nonterminal iterations is at most

$$\frac{d \ln m}{\ln 2 - 1/3}.$$

There is at most one additional successful iteration, which terminates the algorithm. Conditional on any history, the probability of t consecutive unsuccessful iterations is at most 2^{-t}. Hence the conditional expected waiting time for a successful iteration is at most $\sum_{t\geq 0} 2^{-t} = 2$. It follows that the algorithm terminates almost surely and that its total number T of iterations satisfies

$$\mathbb{E}[T] \leq 2\left(1 + \frac{d \ln m}{\ln 2 - 1/3}\right) = \mathcal{O}(d \log m.$$

□

Lemma J.4 (Phase 1 Iteration Bound)**.** *The* HYBRIDCLARKSON *algorithm in Algorithm 8 terminates almost surely and uses $\mathcal{O}(d)$ iterations in expectation. Moreover, the size of the active working set $\mathcal{S}'$ is always at most $\mathcal{O}(d^2\sqrt{s})$.*

Proof. Conditioned on the preceding rounds, $\mathcal{T}$ is fixed and $\mathcal{R}$ is a fresh uniform sample of size $r = \min\{s, \lceil d\sqrt{s}\rceil\}$. Each call to NESTEDMWU terminates almost surely by Lemma J.3 and returns the selected optimizer over $\mathcal{S}' = \mathcal{R} \cup \mathcal{T}$. If $r = s$, this point is globally feasible and the algorithm terminates. Otherwise, Lemma 2.2 gives

$$\mathbb{E}\left[|\mathcal{V}_{global}| \mid \text{preceding rounds}\right] \leq \frac{ds}{r} \leq \sqrt{s}.$$

We call an iteration *successful* if $|\mathcal{V}_{global}| \leq 2d\sqrt{s}$. By Markov's inequality,

$$\Pr\left[|\mathcal{V}_{global}| > 2d\sqrt{s} \mid \text{preceding rounds}\right] \leq \frac{\sqrt{s}}{2d\sqrt{s}} \leq \frac{1}{2}.$$

Thus each iteration succeeds with conditional probability at least $\frac{1}{2}$.

Let $\mathcal{B}^*_{global}$ be a basis preserving the selected global optimizer, with $|\mathcal{B}^*_{global}| \leq d$. If a successful iteration does not terminate, its candidate x^* violates some constraint in $\mathcal{S}$. By Lemma J.1, it must violate at least one constraint in $\mathcal{B}^*_{global}$.

Since x^* is the selected optimizer over $\mathcal{S}' = \mathcal{R} \cup \mathcal{T}$, it satisfies every constraint in $\mathcal{T}$. Thus any violated basis constraint lies in $\mathcal{B}^*_{global} \setminus \mathcal{T}$. The update

$$\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{V}_{global}$$

therefore inserts at least one new basis constraint into $\mathcal{T}$.

Because the basis has at most d constraints, there are at most d successful nonterminal iterations. Once $\mathcal{B}^*_{global} \subseteq \mathcal{T}$, the selected optimizer over any working set containing $\mathcal{T}$ is the selected global optimizer, so the next iteration terminates. Including the terminal iteration, at most $d+1$ successful iterations are needed.

Conditional on any history, the probability of t consecutive unsuccessful iterations is at most 2^{-t}. Thus the conditional expected waiting time for a successful iteration is at most $\sum_{t\geq 0} 2^{-t} = 2$. It follows that the algorithm terminates almost surely and uses at most $2(d+1) = \mathcal{O}(d)$ iterations in expectation.

Finally, each successful nonterminal iteration adds at most $2d\sqrt{s}$ constraints to $\mathcal{T}$. Over at most d such iterations, $|\mathcal{T}| \leq 2d^2\sqrt{s}$. Consequently,

$$|\mathcal{S}'| \leq |\mathcal{T}| + |\mathcal{R}| \leq 2d^2\sqrt{s} + \lceil d\sqrt{s}\rceil = \mathcal{O}(d^2\sqrt{s}),$$

as claimed. □

Lemma J.5. *Suppose the domain Q is a box whose facet inequalities have L-bit coefficients. In the blackboard model, the expected total communication complexity of* HYBRIDCLARKSON *is*

$$O(sd) + \left(L \log n + \log^2 n + \sqrt{s}\right) \log(sd) \cdot \text{poly}(d).$$

Proof. Consider a fixed working set $\mathcal{S}'$ of size m. By Lemma J.3, NESTEDMWU uses $O(d \log m)$ iterations in expectation. Each iteration invokes FINDOPT on at most $6d^2$ distinct sampled servers.

We first account for the fixed lexicographic tie-breaking rule. Implement FINDOPT by successively minimizing $\phi, x_1, \ldots, x_d$, retaining each previously attained minimum as a public equality. Each successive feasible region is a face of the original boxed polytope, so its vertices are vertices of that polytope. By the determinant bound in Appendix I, each such vertex has the form p/q, where $\|p\|_\infty, q \le D := (d!)2^{Ld}$. Coordinate values therefore have $O(\log D)$-bit representations. The value of ϕ at such a vertex has denominator at most $2^L D$ and numerator of magnitude at most $d2^L D$. Consequently, after clearing denominators and rescaling, all added equalities have coefficient precision $O(d(L + \log d))$.

Thus the $d+1$ optimization calls, with $O(d)$ additional public constraints, increase the communication bound of Theorem G.1 only by a factor polynomial in d. Their total expected communication is

$$\big(L \log n + \log^2 n\big) \cdot \mathrm{poly}(d).$$

Publishing the attained scalar values costs an additional $L \cdot \mathrm{poly}(d)$ bits, which is absorbed in this bound. After the coordinate minimizations, the selected point is known to every server from the blackboard.

Feasibility checking against $\mathcal{S}'$ then requires one bit from each of its m servers. The weights are determined by these public violation bits. If sampled indices need to be communicated, writing their indices within $\mathcal{S}'$ costs $O(d^2 \log m)$ bits per iteration, also absorbed below. Hence the expected communication per inner iteration is

$$\big(m + L \log n + \log^2 n\big) \cdot \mathrm{poly}(d).$$

Combining this with Lemma J.3, the expected cost of solving the working set is

$$\big(L \log n + \log^2 n + m\big) \log m \cdot \mathrm{poly}(d).$$

By Lemma J.4, $m = O(d^2\sqrt{s})$ and $\log m = O(\log sd)$. Thus each outer iteration has expected inner-loop cost at most

$$\big(L \log n + \log^2 n + \sqrt{s}\big) \log sd \cdot \mathrm{poly}(d).$$

Its global candidate is already available on the blackboard, so the global feasibility check costs only s additional bits. Announcing the outer sample, if necessary, costs $O(d\sqrt{s} \log s)$ bits, which is absorbed in the preceding bound.

These per-iteration bounds hold uniformly conditional on the preceding history. Applying the expected $O(d)$ outer-iteration bound from Lemma J.4 therefore gives

$$O(sd) + \big(L \log n + \log^2 n + \sqrt{s}\big) \log(sd) \cdot \mathrm{poly}(d),$$

as claimed. □

Putting things together, we have:

Theorem 1.5. *There is an algorithm that solves distributed linear programming using total communication $O(sd) + (L \log n + \log^2 n + \sqrt{s}) \log(sd) \cdot \mathrm{poly}(d)$ in the blackboard model, where each $\mathcal{C}_i$ is an intersection of n_i half-spaces expressed to L bits of precision, $n = \sum n_i$, and ϕ is expressed to L bits of precision.*

K LIMITATIONS AND FUTURE DIRECTIONS

In this section, we briefly outline the theoretical limitations of our current framework and discuss how they motivate future research directions.

High-Dimensional Scaling. Our tightest theoretical bounds depend on the ambient dimension d being small. In particular, the near-optimal expected $O_d(s + L \log(s+1))$ query complexity in the BL-DSO model relies on a nested multiplicative-weights refinement of the active-set framework, with dimension-dependent factors hidden in the notation. Furthermore, our exact optimization bounds rely on a Diophantine simplex-volume bound (Proposition I.2) to reduce the affine dimension of the remaining candidate vertices. Applying these algorithms to extremely high-dimensional spaces,

e.g., modern deep neural network parameter spaces, is currently impractical. Fully resolving the optimal d dependence in the DSO model remains a key open question.

Exact vs. Approximate Optimization. To achieve finite-time bounds for exact linear programming, our work relies heavily on the assumption of explicit bit-precision limits (L) on the inputs. While this is standard in exact linear programming literature (and maps well to standard floating-point architectures in practice), it constitutes a theoretical limitation for arbitrary continuous convex sets. In Section 3.1, we partially mitigate this by providing a continuous algorithm for the convex feasibility problem under a mild volume assumption without bit-precision limits. However, designing purely continuous distributed algorithms that yield ϵ-approximate optimal solutions, without relying on bit-limited Diophantine geometry, is a crucial next step for generalizing the DSO model.

System-Level Network Conditions. Our explicit communication models and empirical simulations track abstract algorithmic complexity, e.g., bits, floats, and synchronization barriers, under the assumption of reliable communication channels. We do not explicitly model physical system-level effects such as Byzantine adversaries, dropped packets, or asynchronous stragglers, which are common in real-world federated learning. Adapting the DSO model to incorporate asynchronous or noisy separation queries is an interesting direction for systems-focused research.