

SPENCER’S THEOREM IN NEARLY INPUT-SPARSITY TIME

VISHESH JAIN, ASHWIN SAH, AND MEHTAAB SAWHNEY

ABSTRACT. A celebrated theorem of Spencer states that for every set system $S_1, \dots, S_m \subseteq [n]$, there is a coloring of the ground set with $\{\pm 1\}$ with discrepancy $O(\sqrt{n \log(m/n + 2)})$. We provide an algorithm to find such a coloring in near input-sparsity time $\tilde{O}(n + \sum_{i=1}^m |S_i|)$. A key ingredient in our work, which may be of independent interest, is a novel width reduction technique for solving linear programs, not of covering/packing type, in near input-sparsity time using the multiplicative weights update method.

1. INTRODUCTION

The celebrated ‘Six standard deviations suffice’ theorem of Spencer [22] says that there is a universal constant $C > 0$ ¹ such that given a set system $S_1, \dots, S_m \subseteq [n] := \{1, \dots, n\}$, there exists a bi-coloring $x := (x_1, \dots, x_n) \in \{\pm 1\}^n$ for which

$$\max_{i \in [m]} \left| \sum_{j \in S_i} x_j \right| \leq C \sqrt{n \log(m/n + 2)}. \quad (1.1)$$

The strength of this result is most readily apparent when $m = n$, in which case the right hand side of (1.1) is $O(\sqrt{n})$ ² whereas a basic application of the probabilistic method only gives the weaker estimate $O(\sqrt{n \log n})$.

Spencer’s proof of this result was non-algorithmic, based on the partial coloring technique of Beck [8]. A different (but still, non-algorithmic) proof, based on convex geometry, was obtained independently by Gluskin [16]. The problem of finding an $x \in \{\pm 1\}^n$, achieving the guarantee of (1.1), in (probabilistic) polynomial time was first solved in a breakthrough work of Bansal [7], by using a random walk guided by the solution to a semi-definite program (SDP)³. A much simpler random-walk based approach was later found by Lovett and Meka [20]. Subsequently, Rothvoss [21] and Eldan and Singh [14] devised randomized polynomial-time algorithms, based on convex geometry, which are also applicable to a generalization of Spencer’s result due to Giannopoulos [15]. Without resorting to fast matrix multiplication (and restricting ourselves here to $m = \Theta(n)$ for ease of presentation), the fastest of these algorithms are those of Lovett–Meka and Eldan–Singh, both running in time $\tilde{O}(n^3)$, where \tilde{O} hides polylogarithmic factors in n . Allowing fast matrix multiplication, the running time of the algorithm of Eldan–Singh is dominated by the time to solve (to sufficient accuracy) a linear program with n variables and $\Theta(n)$ constraints, for which the best-known bound is $\tilde{O}(n^\omega)$ (where $\omega \approx 2.37$ is the *current* best matrix-multiplication exponent [2]) and $\tilde{O}(n^{2+1/18})$ even in the most optimistic case that the matrix-multiplication exponent is 2 and the dual matrix-multiplication exponent is 1 [17].

Sah and Sawhney were supported by NSF Graduate Research Fellowship Program DGE-1745302. Sah was supported by the PD Soros Fellowship.

¹In our notation, Spencer showed that $C < 6/\sqrt{\log 3}$ suffices (when $m = n$), hence the name of the result.

²Hadamard set systems show that this is optimal up to the implicit constant, see [3, Theorem 13.4.1].

³Bansal’s algorithm admits a derandomization. Another deterministic algorithm for Spencer’s theorem was provided by Levy, Ramadas, and Rothvoss [18].

In the Workshop on Discrepancy Theory and Integer Programming in 2018 [13], devising discrepancy minimization algorithms running in near input-sparsity time was suggested as one of the main directions for future research, noting that such algorithms are not known for any of the major results in discrepancy theory, including Spencer’s theorem, the Beck–Fiala theorem [9], or Banaszczyk’s theorem [6]. In fact, all known algorithms for these problems (which produce a coloring with discrepancy at most a universal constant factor more than the best-known existential bounds) employ linear algebraic primitives such as solving a system of linear equations, which suffer from the matrix multiplication bottleneck. We note here that there are recent online algorithms for discrepancy minimization [4, 19] which do not use any such operations and run in input-sparsity time. However, these algorithms are only guaranteed to produce a coloring matching Banaszczyk’s bound up to a polylogarithmic (as opposed to constant) factor. Moreover, in the setting of Spencer’s theorem, Banaszczyk’s bound itself only corresponds to a discrepancy of $O(\sqrt{n \log n})$, which is already attained with high probability by a uniformly random coloring.

In this work, we initiate the study of optimal discrepancy minimization algorithms running in near input-sparsity time, by providing such an algorithm for Spencer’s theorem.

Theorem 1.1. *There exists an absolute constant $C_{1.1} > 0$ and a randomized algorithm Coloring such that the following holds. On input a matrix $A \in \mathbb{R}^{m \times n}$ such that $\|A\|_{1 \rightarrow \infty} \leq 1$, Coloring(A) runs in time $\tilde{O}(\text{nnz}(A) + n)$ and with probability at least $1/2$, outputs a vector $v \in \{\pm 1\}^n$ such that*

$$\|Av\|_{\infty} \leq C_{1.1} \sqrt{n \log(m/n + 2)}.$$

Remark. To see that this recovers (1.1), we let the i^{th} row of A be the indicator vector of S_i and note that $\|A\|_{1 \rightarrow \infty} = \max_{i,j} |a_{i,j}| \leq 1$.

A pleasant feature of our algorithm (see Section 1.1 for an overview) is that it relies not on a new approach to proving Spencer’s theorem, but rather on solving the linear-program of Eldan–Singh in near-input sparsity time by leveraging certain key structural aspects, thereby raising the possibility that the linear algebraic primitives involved in other algorithms for discrepancy minimization may also be implemented much more efficiently. Moreover, during the course of our algorithm, we develop a novel method for solving a certain class of linear-programs (to polylogarithmic relative accuracy) in near-input sparsity time, which may be of independent interest as a rare instance of “width reduction” for linear-programs not of covering/packing type.

1.1. Proof outline. We use the notation in (1.1). For simplicity, we consider the case $m = n$ and $\sum_{i=1}^n |S_i| = \Theta(n^2)$ which already contains most of the ideas; later, we will sketch the modifications needed for the general case. For $C > 0$, we let

$$\Gamma_C := \left\{ x \in [-1, 1]^n : \max_{i \in [m]} \left| \sum_{j \in S_i} x_j \right| \leq C\sqrt{n} \right\}.$$

By a standard reduction (see Theorem 2.2), it suffices to devise an algorithm which runs in time $\tilde{O}(n^2)$ and finds $x \in \Gamma_C$ such that x has at least n/C coordinates which are ± 1 .

Our starting point for doing this is the aforementioned theorem of Eldan–Singh [14] which shows that there is an absolute constant $C > 0$ such that with high probability over the choice of a random Gaussian vector $g \sim \mathcal{N}(0, 1)^{\otimes n}$,

$$x^* := \arg \max_{x \in \Gamma_C} \langle g, x \rangle$$

has n/C coordinates which are ± 1 . In particular, we can find a point with the properties we want by solving the above linear program. In fact, a slight modification of the argument in [21, Section 3] shows that it suffices to solve the linear program to within relative accuracy $1/\text{poly}(n)$ and then

randomly round the approximate maximizer. However, solving the linear program to such small relative error forces us to use high-precision methods for solving linear programs, such as cutting-plane or interior-point methods, for which the matrix-multiplication bottleneck seems rather hard to circumvent. Instead, we rely on first-order optimization methods.

Stability of the linear program: In order to use first-order methods in time $\tilde{O}(n^2)$, we need to show that solving the linear program to within $1/\text{poly}(\log n)$ relative accuracy (and then randomly rounding the approximate solution) suffices. More concretely, we show (Proposition 2.3) that all points achieving at least a $(1 - c_1/\log n)$ -factor of the optimum have $\Omega(n)$ coordinates with absolute value at least $1 - c_2/\log n$; these coordinates can then be randomly rounded to have absolute value 1, while only incurring additional discrepancy $O(\sqrt{n})$. Our proof of this shares some similarities to [14, 21], but ultimately relies on a different phenomenon: we use a supersaturation version of Spencer’s theorem due to Spencer [22] as well as Gaussian concentration for Lipschitz functions to show (Lemma 3.4) that the expected value of the linear program is $(\sqrt{2/\pi} - \delta_C)n$, where $\delta_C \rightarrow 0$ as $n \rightarrow \infty$, which further enables us to show that the “derivative” of the map $C \mapsto \mathbb{E}[\max_{x \in \Gamma_C} \langle g, x \rangle]$ is sufficiently small, in a suitably averaged sense, for sufficiently large C (Corollary 3.5). This gives us the desired conclusion since, for sufficiently large C , if there were a point within at least a $(1 - c_1/\log n)$ -factor of $\mathbb{E}[\max_{x \in \Gamma_C} \langle g, x \rangle]$ with only sublinear coordinates with absolute value at least $1 - c_2/\log n$, then it turns out we could find a point $x' \in \Gamma_{C(1+c_2/\log n)}$ with $\langle g, x' \rangle - \mathbb{E}[\max_{x \in \Gamma_{C(1+c_2/\log n)}} \langle g, x \rangle] = \tilde{\Omega}(n)$. However, by Gaussian concentration, this can only happen with probability at most $\exp(-\tilde{\Omega}(n))$ in the choice of g .

The Multiplicative Weights Update (MWU) framework: Our task is now reduced to solving the linear program, which we rewrite in a more convenient form as a feasibility/search problem: for given $C > 0$, find x such that

$$\begin{aligned} x &\in [-1, 1]^n \\ -\langle g/\sqrt{n}, x \rangle &\leq -\text{OPT}_C \\ Ax &\leq C\sqrt{n} \cdot \mathbb{1}, \end{aligned} \tag{1.2}$$

where $\text{OPT}_C := \max_{x \in \Gamma_C} \langle g, x \rangle / \sqrt{n}$ ⁴ and $A \in \mathbb{R}^{2n \times n}$ with rows $A_{2i} := \mathbb{1}_{S_i}$, $A_{2i+1} := -\mathbb{1}_{S_i}$. In fact, by the stability of the linear program, it suffices to find $x \in [-1, 1]^n$ which satisfies the remaining inequalities up to an additive term of order $\sqrt{n}/\log n$ on the right hand side. By the usual multiplicative weights update method for solving linear programs (see, e.g., [5]), this can be done using iterations consisting of solving a *single* linear inequality over the unit cube: find $x' \in [-1, 1]^n$ s.t.

$$-\rho_0 \langle g/\sqrt{n}, x' \rangle + \sum_{i=1}^{2n} \rho_i \langle A_i, x' \rangle \leq -\rho_0 \text{OPT}_C + (1 - \rho_0)C\sqrt{n} + \sqrt{n}/\log n, \tag{1.3}$$

where $\rho_0, \rho_1, \dots, \rho_{2n}$ is a given probability distribution on $[2n + 1]$ (corresponding to the “weights of the experts” in the MWU framework). Note that such an x , if it exists, can be found in time $O(n^2)$ using a greedy coordinate-by-coordinate strategy. Therefore, if $\tilde{O}(1)$ iterations of MWU were sufficient, then we would be done

⁴We do not know OPT_C a priori; however, by using a standard binary search procedure, we may assume access to a sufficiently good approximation, and we will ignore the distinction in this sketch.

The number of iterations required by MWU to give a solution with the required accuracy is $\tilde{O}(w^2/n)$, where w is the “width” of the procedure measuring the maximum violation of any constraint by the intermediate solutions x' . In our case, it is easy to see that

$$w = O(\sqrt{n}) + \max_{x'} \langle g/\sqrt{n}, x' \rangle + \max_{x'} \max_{i \in [2n]} |\langle A_i, x' \rangle| = O(\sqrt{n}) + \max_{x'} \max_{i \in [2n]} |\langle A_i, x' \rangle|,$$

where x' ranges over the solutions to the single linear inequality output by different iterations of MWU and where we have used that $\max_{x' \in [-1, 1]^n} |\langle g/\sqrt{n}, x' \rangle| = O(\sqrt{n})$ as long as $\|g\|_2 = O(\sqrt{n})$, which holds except with exponentially small probability over the choice of g . Therefore, if it were the case that

$$\max_{x'} \max_{i \in [2n]} |\langle A_i, x' \rangle| = \tilde{O}(\sqrt{n}),$$

then $\tilde{O}(w^2/n) = \tilde{O}(1)$, as required. Unfortunately, with the greedy coordinate-by-coordinate strategy for solving the single linear inequality on the cube, the width could potentially be of order n , so that $\tilde{O}(w^2/n) = \tilde{O}(n)$ and we get a total running time of $\tilde{O}(n^3)$, as for previously known algorithms.

Width reduction: To overcome this obstacle, we develop a novel “width reduction” technique (see the proof that [Proposition 4.2](#) implies [Proposition 4.1](#)), which guarantees that the width is $\tilde{O}(1)$ while blowing up the cost of each iteration by a factor of at most $\tilde{O}(1)$. In the optimization literature, width reduction techniques have been famously used to solve non-negative linear programs in near input-sparsity time (see, e.g., [\[1, Table 1\]](#)). However, our linear program is not of this form and our technique is completely different. In each iteration, instead of considering the simple linear inequality mentioned earlier, consider the following convex program: find $x' \in [-1, 1]^n$ minimizing

$$-\rho_0 \langle g/\sqrt{n}, x' \rangle + \sum_{i=1}^{2n} \rho_i \langle A_i, x' \rangle + \frac{1}{\text{poly}(\log n)} \|Ax'\|_\infty,$$

possibly up to an additive error of $\sqrt{n}/(2 \log n)$. This program has a few crucial properties:

- By the feasibility of [\(1.2\)](#), it follows that the optimum value is at most the right hand side of [\(1.3\)](#). Let x' denote the (approximate) solution returned by the convex program. If $\|Ax'\|_\infty = \tilde{O}(\sqrt{n})$, then the point x' is a solution to [\(1.3\)](#) with width $\tilde{O}(\sqrt{n})$, which suffices for our purpose.
- By testing at $x' = 0$, we see that the optimum value is always non-positive. In particular, any minimizer $x' \in [-1, 1]^n$ satisfies

$$\sum_{i=1}^{2n} \rho_i \langle A_i, x' \rangle + \frac{1}{\text{poly}(\log n)} \|Ax'\|_\infty \leq \|g\|_2.$$

Therefore, assuming $\|g\|_2 = O(\sqrt{n})$, if the minimizer x' satisfies $\|Ax'\|_\infty \gg \sqrt{n} \text{poly}(\log n)$ then it must be that

$$\sum_{i=1}^{2n} \rho_i \langle A_i, x' \rangle \ll -\sqrt{n} \text{poly}(\log n),$$

in which case $x'' = \tilde{O}(x'/\|Ax'\|_\infty)$ is readily seen to satisfy [\(1.3\)](#) with width $\tilde{O}(1)$.

- The program can be written as a linear min-max program (or ℓ_∞ - ℓ_1 matrix game):

$$\min_{x' \in [-1, 1]^n} \max_{y \in \Delta_{2n+1}} y_0 \rho_0 \left(\langle g/\sqrt{n}, x' \rangle + \sum_{i=1}^{2n} \rho_i \langle A_i, x' \rangle \right) + \sum_{i=1}^{2n} y_i \langle A_i, x' \rangle / \text{poly}(\log n),$$

where $\Delta_{2n+1} \subset \mathbb{R}^{2n+1}$ denotes the unit simplex consisting of probability distributions on $[2n + 1]$, which one can try to solve using a saddle-point mirror descent scheme. At this

point, it is natural to think that the correct geometry on $[-1, 1]^n$ is given by the ∞ -norm, so that one runs into the usual problem of the lack of an $\tilde{\Omega}(1)$ -strongly convex mirror map on $[-1, 1]^n$. However, we can take advantage of the fact that $\|A_i\|_2 = O(\sqrt{n})$ in Spencer’s problem (and that projecting onto $[-1, 1]^n$ with respect to the Euclidean norm is easy) by viewing $[-1, 1]^n$ as a subset of the ℓ_2 ball of radius \sqrt{n} and using the sublinear primal-dual framework of Clarkson, Hazan, and Woodruff [12] to show that the linear min-max program can indeed be solved to desired accuracy in time $\tilde{O}(n^2)$ (Proposition 5.1).

Running in near input-sparsity time: So far, we have assumed that $m = n$ and $\sum_{i=1}^n |S_i| = \tilde{O}(n^2)$. The same discussion extends to handle the case of general $m \leq n^2$ ⁵ under the additional assumption that the set system is “dense” i.e. $\sum_{i=1}^m |S_i| = \tilde{O}(mn)$. To remove the density assumption, we require a few additional ingredients.

- First, we isolate variables which appear in at most $n/\text{poly}(\log n)$ sets and color them using the input-sparsity time online algorithm for Banaszczyk’s theorem due to Alweiss, Liu, and Sawhney [4] (Appendix A). As mentioned earlier, this algorithm loses a polylogarithmic factor in the discrepancy guarantee, compared to Banaszczyk’s theorem. However, since we are coloring only those variables which are in at most $n/\text{poly}(\log n)$ sets, such a loss is acceptable.
- At this point, each of the n_1 remaining variables appears in at least $n/\text{poly}(\log n)$ sets and at most m sets. Once again, we may assume that $m \leq n_1^2$, else a uniformly random coloring suffices. Let the total number of non-zero entries in this restricted incidence matrix be N , so that $n_1 n/\text{poly}(\log n) \leq N \leq n_1 m$. We isolate the at most $n_1/\log n$ variables which are present in at least $(\log n) \cdot N/n_1$ sets and color them uniformly at random. Note that a uniformly random coloring loses a factor of $\sqrt{\log n_1}$ compared to Spencer’s bound, but this again acceptable, since we are only coloring at most $n_1/\log n$ variables.
- Finally, all the remaining variables have the property of being present in at most $k = (\log n) \cdot N/n_1$ sets each and a careful implementation of [12] using a suitable (but fairly simple) data structure shows that the linear min-max program, restricted to such variables, can be solved in time $\tilde{O}(kn_1 + n_1^2) = \tilde{O}(N) = \tilde{O}(\sum_{i=1}^m |S_i|)$, as desired.

2. PRELIMINARY REDUCTIONS

In this section, we formally record a couple of preliminary reductions, which allow us to deduce Theorem 1.1 from a similar statement about appropriate ‘partial colorings’ for input matrices A with $\tilde{\Omega}(n^2)$ non-zero entries.

First, by using an input-sparsity time online algorithm for the Komlós conjecture (with polylogarithmic losses) due to Alweiss, Liu, and the third author [4] (with an alternate proof by Liu and the last two authors [19]) on those variables which appear in $O(n/(\log n)^2)$ sets, it suffices to prove Theorem 1.1 with an extra additive n^2 in the running time.

Theorem 2.1. *There exists an absolute constant $C_{2.1} > 0$ and a randomized algorithm Dense-Coloring such that the following holds. On input a matrix $A \in \mathbb{R}^{m \times n}$ such that $\|A\|_{1 \rightarrow \infty} \leq 1$, Dense-Coloring(A) runs in time $\tilde{O}(\text{nnz}(A) + n^2)$ and with probability at least $1/2$, outputs a vector $v \in \{\pm 1\}^n$ such that*

$$\|Av\|_\infty \leq C_{2.1} \sqrt{n \log(m/n + 2)}.$$

The proof that Theorem 2.1 implies Theorem 1.1 is presented in Appendix A.

Next, in order to prove Theorem 2.1, it suffices to prove the following statement about colorings valued in $[-1, 1]^n$ with linearly many coordinates colored by ± 1 .

⁵If $m \geq n^2$, then a uniformly random coloring succeeds with high probability.

Theorem 2.2. *There exists an absolute constant $C_{2.2} > 0$ and a randomized algorithm Partial-Coloring such that the following holds. On input*

- a diagonal matrix $\Lambda \in \mathbb{R}^{n \times n}$ with $\|\Lambda\|_{1 \rightarrow \infty} \leq 1$, and
- a matrix $A \in \mathbb{R}^{m \times n}$ with $\|A\|_{1 \rightarrow \infty} \leq 1$,

Partial-Coloring(A, Λ) runs in time $\tilde{O}(\text{nnz}(A) + n^2)$ and with probability at least $1/2$ outputs a vector $v \in [-1, 1]^n$ such that

- $\sum_{i \in [n]} \mathbb{1}_{|v_i|=1} \geq \frac{C_{2.2}^{-1}}{2.2} \cdot n$, and
- $\|A\Lambda v\|_\infty \leq C_{2.2} \sqrt{n \log(m/n + 2)}$.

The deduction of [Theorem 2.1](#) from [Theorem 2.2](#) is, by now, standard, and essentially identical to that in Rothvoss [[21](#)]; we include the details in [Appendix A](#) for the sake of completeness.

[Theorem 2.2](#) follows from the following pair of propositions.

Proposition 2.3. *There exist absolute constants $C_{2.3}, \eta_{2.3} > 0$ such that the following holds. Given $A \in \mathbb{R}^{m \times n}$ such that $\|A\|_{1 \rightarrow \infty} \leq 1$, independently sample $g \sim \mathcal{N}(0, 1)^{\otimes n}$ and $C_{\text{Alg}} \sim \text{Unif}([C_{2.3}, 2C_{2.3}])$, and let*

$$\Gamma_{A, C_{\text{Alg}}} := \{x \in \mathbb{R}^n : \|Ax\|_\infty \leq C_{\text{Alg}} \sqrt{n \log(m/n + 2)} \wedge \|x\|_\infty \leq 1\}.$$

Then the following hold:

- with probability at least $1 - \exp(-\Omega(n))$, we have that

$$\|g\|_2 \leq 2\sqrt{n},$$

- with probability at least $4/5$, for any $x \in \Gamma_{A, C_{\text{Alg}}}$ such that

$$\langle g, x \rangle / \sqrt{n} \geq \sup_{y \in \Gamma_{A, C_{\text{Alg}}}} \langle g, y \rangle / \sqrt{n} - \varepsilon \eta_{2.3} \sqrt{n},$$

we have that

$$\sum_{j \in [n]} \mathbb{1}_{|x_j| \geq 1 - \varepsilon} \geq \eta_{2.3} n,$$

where $\varepsilon = 1/\log n$.

Proposition 2.4. *For any $C \leq \sqrt{\log n}/3$ and $n \geq 100$, there exists a randomized algorithm Solve such that the following holds. On input*

- a matrix $A \in \mathbb{R}^{m \times n}$ with $m \leq n^2$, $\|A\|_{1 \rightarrow \infty} \leq 1$, and $\max_{i \in [n]} |\text{supp}(Ae_i)| \leq k$, and
- a vector $v \in \mathbb{R}^n$ such that $\|v\|_2 \leq 2\sqrt{n}$,

Solve(A, v) runs in time $\tilde{O}(kn + n^2)$ and with probability at least $99/100$, outputs

$$z \in \Gamma_{A, C} := \{x : \|x\|_\infty \leq 1 \wedge \|Ax\|_\infty \leq C \sqrt{n \log(m/n + 2)}\}$$

such that

$$\langle v, z \rangle / \sqrt{n} \geq \sup_{y \in \Gamma_{A, C}} \langle v, y \rangle / \sqrt{n} - \varepsilon \sqrt{n},$$

where $\varepsilon = 1/(\log n)^2$.

Remark. Since $0 \in \Gamma_{A, C} \subseteq \{y : \|y\|_2 \leq \sqrt{n}\}$, we have

$$0 \leq \sup_{y \in \Gamma_{A, C}} \langle v, y \rangle / \sqrt{n} \leq \sup_{\|y\|_2 \leq \sqrt{n}} \langle v, y \rangle / \sqrt{n} \leq 2\sqrt{n}. \quad (2.1)$$

We conclude this section by showing how to deduce [Theorem 2.2](#) from the above propositions.

Proof of Theorem 2.2. By replacing A by $A\Lambda$ (which can be computed in time $O(\text{nnz}(A) + n)$ and satisfies the same guarantee $\|A\Lambda\|_{1 \rightarrow \infty} \leq 1$), it suffices to consider the case when $\Lambda = I_n$. We may also assume that $m \leq n^2$ since if $m > n^2$, a direct application of Bernstein's concentration inequality and the union bound shows that a uniformly random assignment $v \sim \{\pm 1\}^n$ satisfies the conclusion of Theorem 2.2 with high probability.

Now, given $A \in \mathbb{R}^{m \times n}$ with $\|A\|_{1 \rightarrow \infty} \leq 1$ and $\Lambda = I_n$, define

$$\mathcal{C}_{\text{Heavy}} := \{i : \text{supp}(Ae_i) \geq (\log n) \text{nnz}(A)/n\}.$$

We decompose A into A_1 and A_2 with columns indexed by $\mathcal{C}_{\text{Heavy}}$ and $[n] \setminus \mathcal{C}_{\text{Heavy}}$ respectively.

Note that $A_1 \in \mathbb{R}^{m \times \mathcal{C}_{\text{Heavy}}}$, and by Markov's inequality, $|\mathcal{C}_{\text{Heavy}}| \leq n/\log n$. Therefore, by Bernstein's inequality and a union bound, a uniformly random $v_1 \in \{\pm 1\}^{\mathcal{C}_{\text{Heavy}}}$ satisfies $\|A_1 v_1\|_\infty \leq C\sqrt{n}$ with probability at least $9/10$, for a sufficiently large absolute constant $C > 0$.

Thus it suffices to find a suitable coloring for $A_2 \in \mathbb{R}^{m \times ([n] \setminus \mathcal{C}_{\text{Heavy}})}$. By Propositions 2.3 and 2.4, in time $\tilde{O}(n \cdot ((\log n) \text{nnz}(A)/n) + n^2) = \tilde{O}(\text{nnz}(A) + n^2)$ and with probability at least $3/4$, we can find $v'_2 \in [-1, 1]^{[n] \setminus \mathcal{C}_{\text{Heavy}}}$ such that

- $\|A_2 v'_2\|_\infty \leq 2C_{2.3} \sqrt{n \log(2m/n + 2)}$, and
- $T := \{j : |(v'_2)_j| \geq 1 - 2/\log n\}$ satisfies $|T| \geq \eta_{2.3} n/2$.

Let v_2 be the random vector obtained from v'_2 by randomly rounding each coordinate in T independently to $\{\pm 1\}$ in a manner such that $\mathbb{E}[(v_2)_j] = (v'_2)_j$ for all $j \in [T]$. Then, by Bernstein's inequality and the union bound, it follows that with high probability,

- $\|A_2 v_2\|_\infty \leq 4C_{2.3} \sqrt{n \log(2m/n + 2)}$, and
- $T := \{j : |(v_2)_j| = 1\}$ satisfies $|T| \geq \eta_{2.3} n/4$,

so that $v = (v_1, v_2)$ can be obtained in time $\tilde{O}(\text{nnz}(A) + n^2)$, and with probability at least $1/2$, satisfies the conclusion of Theorem 2.2. \square

3. STABILITY OF THE LINEAR PROGRAM

The goal of this section is to prove Proposition 2.3, which states that the linear program of Eldan and Singh [14] is logarithmically stable, in the sense that all points with objective value within a $(1 - c_1/\log n)$ -factor of the optimum have linearly many coordinates with absolute value at least $1 - c_2/\log n$.

As in the statement of the lemma, fix a matrix $A \in \mathbb{R}^{m \times n}$ such that $\|A\|_{1 \rightarrow \infty} \leq 1$, and for $C > 0$, define

$$\Gamma_{A,C} := \{x \in \mathbb{R}^n : \|Ax\|_\infty \leq C\sqrt{n \log(m/n + 2)} \wedge \|x\|_\infty \leq 1\}.$$

Furthermore, define

$$\text{OPT}_A(C) := \mathbb{E}_{g \sim \mathcal{N}(0, I_n)} \left[\max_{y \in \Gamma_{A,C}} \langle g, y \rangle \right] / \sqrt{n}. \quad (3.1)$$

Our proof of Proposition 2.3 requires a few ingredients. First, we record a trivial upper bound on $\text{OPT}_A(C)$.

Lemma 3.1. *Fix $A \in \mathbb{R}^{m \times n}$ such that $\|A\|_{1 \rightarrow \infty} \leq 1$. For any $C > 0$,*

$$\text{OPT}_A(C) \leq \sqrt{2n/\pi}.$$

Proof. Since $\Gamma_{A,C} \subseteq [-1, 1]^n$, we have

$$\text{OPT}_A(C) \leq \mathbb{E}_{g \sim \mathcal{N}(0, I_n)} \left[\max_{\|y\|_\infty \leq 1} \langle g, y \rangle \right] / \sqrt{n} = \sqrt{n} \mathbb{E}_{g \sim \mathcal{N}(0, 1)} [|g|] = \sqrt{2n/\pi}. \quad \square$$

Next, we record a consequence of Spencer's proof of Spencer's Theorem [22].

Lemma 3.2 ([22, Theorem 12]). *Fix $A \in \mathbb{R}^{m \times n}$ such that $\|A\|_{1 \rightarrow \infty} \leq 1$ and let $C > 0$. There exists a constant $\delta_C > 0$ (depending only on C and independent of A) such that*

$$|\Gamma_{A,C} \cap \{\pm 1\}^n| \geq (2 - \delta_C)^n,$$

where $\delta_C \rightarrow 0$ as $C \rightarrow \infty$.

Remark. In [22, Theorem 12], the corresponding result is only stated for $m = n$. A trivial modification of the proof however immediately gives the desired result.

We will also use concentration of Lipschitz functions with respect to the Gaussian measure.

Lemma 3.3 (see, e.g., [14, Theorem 2.2]). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be L -Lipschitz with respect to the Euclidean norm. Then*

$$\mathbb{P}_{g \sim \mathcal{N}(0,1)^{\otimes n}}[|f(g) - \mathbb{E}[f(g)]| \geq t] \leq 2 \exp(-t^2/(2L^2)).$$

The previous two lemmas allow us to show that for sufficiently large C , the trivial upper bound in Lemma 3.1 is close to sharp.

Lemma 3.4. *Fix $A \in \mathbb{R}^{m \times n}$ such that $\|A\|_{1 \rightarrow \infty} \leq 1$ and let $C > 0$. There exists a constant $\delta'_C > 0$ (depending only on C and independent of A) such that*

$$\text{OPT}_A(C) \geq \sqrt{2n/\pi} - \delta'_C \sqrt{n},$$

where $\delta'_C \rightarrow 0$ as $C \rightarrow \infty$.

Proof. For $g \sim \mathcal{N}(0, I_n)$, note that $\text{sgn}(g) \in \{\pm 1\}^n$ is distributed uniformly on $\{\pm 1\}^n$. Furthermore if $\text{sgn}(g) \in \Gamma_{A,C}$, then

$$\max_{y \in \Gamma_{A,C}} \langle g, y \rangle \geq \langle g, \text{sgn}(g) \rangle = \|g\|_1.$$

In particular,

$$\mathbb{P}[\max_{y \in \Gamma_{A,C}} \langle g, y \rangle - \|g\|_1 \geq 0] \geq \mathbb{P}[\text{sgn}(g) \in \Gamma_{A,C}] \geq \exp(-2\delta_C n),$$

where δ_C is as in Lemma 3.2.

Hence, since $g \mapsto \max_{y \in \Gamma_{A,C}} \langle g, y \rangle / \sqrt{n} - \|g\|_1 / \sqrt{n}$ is 2-Lipschitz with respect to the Euclidean norm, it follows from Lemma 3.3 that we must have

$$\mathbb{E}[\max_{y \in \Gamma_{A,C}} \langle g, y \rangle / \sqrt{n} - \|g\|_1 / \sqrt{n}] \geq -8(\delta_C)^{1/2} \sqrt{n}$$

for sufficiently large n . The desired result now follows by the linearity of expectation. \square

As a corollary, we deduce the following stability result for $\text{OPT}_A(C)$ with respect to C .

Corollary 3.5. *There exists an absolute constant $c_{3.5} > 0$ and a non-increasing function $C_{3.5} : [0, 1] \rightarrow \mathbb{R}^{>0}$ for which the following holds. For any $0 \leq \varepsilon \leq c_{3.5}$, $\delta \in [0, 1]$, and $C \geq C_{3.5}(\delta)$,*

$$\mathbb{P}_{C' \sim \text{Unif}[C, 2C]}[\text{Opt}_A(C' + C \cdot \varepsilon) - \text{Opt}_A(C') \leq \delta \cdot \varepsilon \sqrt{n}] \geq 5/6.$$

Remark. It is an interesting problem in convex geometry to determine whether the Lipschitz constant of $C \mapsto \text{Opt}_A(\exp(C))$ is sufficiently small for all C sufficiently large; our proof of Corollary 3.5 shows that this is true in an appropriate averaged sense.

Proof. For $0 \leq i \leq \lfloor 1/(9\varepsilon) \rfloor$, consider $x_i := C + C \cdot i \cdot (9\varepsilon)$. Note that for $C' \in [C + 9iC\varepsilon, C + (9i + 8)C\varepsilon]$, we have that $\text{OPT}_A(C' + \varepsilon) - \text{OPT}_A(C') \leq \text{OPT}_A(x_{i+1}) - \text{OPT}_A(x_i)$. Moreover,

$$\text{Opt}_A(C + \lfloor 1/(9\varepsilon) \rfloor \cdot (9\varepsilon)) - \text{Opt}_A(C) = \sum_{i=0}^{\lfloor 1/(9\varepsilon) \rfloor - 1} \text{Opt}_A(x_{i+1}) - \text{Opt}_A(x_i)$$

$$\geq \delta \cdot \varepsilon \sqrt{n} \cdot \#\{i : \text{Opt}_A(x_{i+1}) - \text{Opt}_A(x_i) \geq \delta \cdot \varepsilon \sqrt{n}\}.$$

By [Lemma 3.4](#), the left hand side is at most $\delta'_C \sqrt{n}$, so that

$$\#\{i : \text{Opt}_A(x_{i+1}) - \text{Opt}_A(x_i) \geq \delta \cdot \varepsilon \sqrt{n}\} \leq \delta'_C \delta^{-1} \varepsilon^{-1}.$$

Therefore,

$$\begin{aligned} \mathbb{P}_{C' \sim \text{Unif}[C, 2C]}[\text{Opt}_A(C' + C \cdot \varepsilon) - \text{Opt}_A(C')] &\leq \delta \cdot \varepsilon \sqrt{n} \\ &\geq (8\varepsilon) \cdot (\lfloor 1/(9\varepsilon) \rfloor) - \#\{i : \text{Opt}_A(x_{i+1}) - \text{Opt}_A(x_i) \geq \delta \cdot \varepsilon \sqrt{n}\} \\ &\geq (8\varepsilon) \cdot \lfloor 1/(9\varepsilon) \rfloor - 8\delta'_C \delta^{-1} \geq 5/6, \end{aligned}$$

provided ε is small and $C_{3.5}(\delta)$ is chosen so that $\delta'_{C_{3.5}}$ is sufficiently small compared to δ . \square

We are now in position to prove [Proposition 2.3](#).

Proof of Proposition 2.3. The first bullet point follows immediately from [Lemma 3.3](#), upon noting that $\mathbb{E}[\|g\|_2] \leq (\mathbb{E}[\|g\|_2^2])^{1/2} = \sqrt{n}$ and that $g \mapsto \|g\|_2$ is a 1-Lipschitz function of the Euclidean norm.

We proceed to the proof of the second bullet point. Recall that $\varepsilon = 1/\log n$ and let η be a sufficiently small constant to be chosen later. Given η , we choose C sufficiently large (according to [Corollary 3.5](#)) so that C_{Alg} satisfies $\text{Opt}_A(C_{\text{Alg}} + C_{\text{Alg}}\varepsilon) - \text{Opt}_A(C_{\text{Alg}}) \leq \varepsilon \eta \sqrt{n}$ with probability at least $5/6$. For the remainder of the proof, we fix C, C_{Alg} satisfying this guarantee. Consider the random quantity

$$\min_{|S|=(1-\eta)n} \max_{\substack{y \in \Gamma_{A, C_{\text{Alg}}} \\ \text{supp}(y) \in S}} \langle g, y \rangle / \sqrt{n}.$$

For a fixed S , we have that

$$g \mapsto \max_{\substack{y \in \Gamma_{A, C_{\text{Alg}}} \\ \text{supp}(y) \in S}} \langle g, y \rangle / \sqrt{n}$$

is 1-Lipschitz with respect to the Euclidean norm, and since C is sufficiently large, we have by [Lemma 3.4](#) that

$$\mathbb{E} \left[\max_{\substack{y \in \Gamma_{A, C_{\text{Alg}}} \\ \text{supp}(y) \in S}} \langle g, y \rangle / \sqrt{n} \right] \geq \sqrt{n}/2.$$

Therefore, by [Lemma 3.3](#) and a union bound over the $\binom{n}{\eta n} \leq 2^{n/100}$ choices for S , it follows that with probability at least $1 - \exp(-\Omega(n))$ over the choice of g ,

$$\min_{|S|=(1-\eta)n} \max_{\substack{y \in \Gamma_{A, C_{\text{Alg}}} \\ \text{supp}(y) \in S}} \langle g, y \rangle \geq \sqrt{n}/4. \quad (3.2)$$

Moreover, by our choice of C_{Alg} and [Lemma 3.3](#), we have that

$$\mathbb{P} \left[\max_{y' \in \Gamma_{A, C_{\text{Alg}} + C_{\text{Alg}}\varepsilon}} \langle g, y' \rangle / \sqrt{n} - \max_{y \in \Gamma_{A, C_{\text{Alg}}}} \langle g, y \rangle / \sqrt{n} < \varepsilon \sqrt{n}/8 \right] \geq 1 - \exp(-\Omega(\varepsilon^2 n)). \quad (3.3)$$

To conclude the proof, we show that on the events appearing in [\(3.2\)](#) and [\(3.3\)](#) (which simultaneously hold with probability $1 - 2\exp(-\Omega(\varepsilon^2 n))$), and for $\eta \leq 1/16$, there cannot exist a vector $x \in \Gamma_{A, C_{\text{Alg}}}$ such that

$$\langle g, x \rangle / \sqrt{n} - \sup_{y \in \Gamma_{A, C_{\text{Alg}}}} \langle g, y \rangle / \sqrt{n} \geq -\varepsilon \eta \sqrt{n} \quad \text{and} \quad \sum_{j \in [n]} \mathbb{1}_{|x_j| \geq 1-\varepsilon} \leq \eta n.$$

Indeed, suppose for contradiction that such a vector x exists. Let S be the set of coordinates where x has magnitude less than $1 - \varepsilon$. By (3.2), there exists $z \in \Gamma_{A, C_{\text{Alg}}}$ such that

$$\langle g, z \rangle / \sqrt{n} \geq \sqrt{n}/4.$$

Consider the vector $z' = x + \varepsilon \cdot z \in \Gamma_{A, C_{\text{Alg}} + C_{\text{Alg}}\varepsilon}$. We have that

$$\langle g, z' \rangle / \sqrt{n} - \max_{y \in \Gamma_{A, C_{\text{Alg}}}} \langle g, y \rangle / \sqrt{n} \geq \varepsilon \sqrt{n}/4 - \varepsilon \eta \sqrt{n} \geq \varepsilon \sqrt{n}/8,$$

and hence,

$$\max_{y' \in \Gamma_{A, C_{\text{Alg}} + C_{\text{Alg}}\varepsilon}} \langle g, y' \rangle / \sqrt{n} - \max_{y \in \Gamma_{A, C_{\text{Alg}}}} \langle g, y \rangle / \sqrt{n} \geq \varepsilon \sqrt{n}/8,$$

which cannot happen on the event appearing in (3.3). \square

4. REDUCTION TO A MINIMAX PROBLEM

The remainder of this paper is devoted to establishing Proposition 2.4. Since we only need to solve the linear program in Proposition 2.4 to $(1 - 1/\text{poly}(\log n))$ -relative error, it is natural to consider a first-order method for solving linear programs, such as using the multiplicative weights update (MWU) method (see [5] for an excellent introduction). Unfortunately, one immediately runs into the issue that the so-called width of the linear program can potentially be $\Theta(n)$, so that the MWU-based solver takes time $\tilde{O}(n^3)$. To overcome this obstacle, we introduce a novel method of ‘width reduction’, which takes advantage of the structure of our linear program.

In the next section, we will show how to implement the following key subroutine which, combined with the standard MWU approach, proves Proposition 2.4.

Proposition 4.1. *For any $C \leq \sqrt{\log n}/3$ and $n \geq 100$, there exists a randomized algorithm Regularized-Solve for which the following holds. On input:*

- a matrix $A \in \mathbb{R}^{m \times n}$ with $m \leq n^2$, $\|A\|_{1 \rightarrow \infty} \leq 1$, and $\max_{i \in [n]} |\text{supp}(Ae_i)| \leq k$,
- $\rho_0 \in \mathbb{R}^{\geq 0}$, $\rho_+, \rho_- \in (\mathbb{R}^{\geq 0})^m$ with $\rho_0 + \|\rho_+\|_1 + \|\rho_-\|_1 = 1$,
- $v \in \mathbb{R}^n$ with $\|v\|_2 \leq 2\sqrt{n}$,
- $\Lambda \in [0, 2\sqrt{n}]$,

Regularized-Solve($A, \rho_0, \rho_+, \rho_-, v, \Lambda$) runs in time $\tilde{O}(kn + n^2)$, and with probability at least $1 - 1/\sqrt{n}$, outputs either:

- a certificate that the program

$$\sup_{x \in \Gamma_{A, C}} \langle x, v \rangle \geq \Lambda \sqrt{n}$$

is not feasible, where

$$\Gamma_{A, C} := \{x \in \mathbb{R}^n : \|x\|_\infty \leq 1 \wedge \|Ax\|_\infty \leq C\sqrt{n \log(m/n + 2)}\}, \quad \text{or}$$

- a point $x \in [-1, 1]^n$ satisfying $\|Ax\|_\infty \leq 10\sqrt{n}(\log n)^4$ and

$$\begin{aligned} & -\rho_0 v^T x / \sqrt{n} + \rho_+^T Ax - \rho_-^T Ax \\ & \leq -\rho_0 \Lambda + (C\sqrt{n \log(m/n + 2)})(\|\rho_+\|_1 + \|\rho_-\|_1) + \sqrt{n}/(\log n)^3 \end{aligned}$$

Remark. Note that it is easy to compute $\arg \min_{x \in [-1, 1]^n} -\rho_0 v^T x + \rho_+^T Ax - \rho_-^T Ax$, which coincides with $-\text{sgn}(-\rho_0 v^T / \sqrt{n} + \rho_+^T A - \rho_-^T A)$, in time $O(\text{nnz}(A) + n)$. The content of this proposition is that we can efficiently find an approximate optimizer x with $\|Ax\|_\infty = \tilde{O}(\sqrt{n})$, which is best possible up to logarithmic factors.

Proof of Proposition 2.4 given Proposition 4.1. We use the notation in [5, Section 3.3]. For convenience of notation, let $C_{m,n} := C\sqrt{n\log(m/n+2)}$. In order to prove Proposition 2.4, it suffices to solve the feasibility program

$$\exists? x \in [-1, 1]^n : \quad \frac{v^T x}{\sqrt{n}} \geq \Lambda; \quad Ax \geq -C_{m,n}\mathbb{1}; \quad -Ax \geq -C_{m,n}\mathbb{1},$$

where $\Lambda := \sup_{y \in \Gamma_{A,C}} \langle v, y \rangle / \sqrt{n}$ ⁶, up to an additive error of $\varepsilon = \sqrt{n}/(\log n)^3$ on the right hand side for each of the three inequalities. Indeed, given such a point x , it is readily seen that $z := (1 - 1/(\log n)^3) \cdot x$ satisfies the conclusion of Proposition 2.4.

For this feasibility program, we begin by noting that Proposition 4.1 provides an (ℓ, ρ) -bounded ORACLE, in the sense of [5, Definition 3.2], with $\ell = \rho = 20\sqrt{n}(\log n)^4$, and with probability at least $1 - 1/\sqrt{n}$. Indeed, the point $x \in [-1, 1]^n$ output by Proposition 4.1 satisfies

$$\max_{i \in [m]} |A_i x - C_{m,n}| \leq \|Ax\|_\infty + C_{m,n} \leq 20\sqrt{n}(\log n)^4,$$

where the final inequality holds since $m \leq n^2$ and $C \leq \sqrt{\log n}$. Also,

$$|\langle v, x \rangle / \sqrt{n} - \Lambda| \leq \Lambda + \|v\|_2 \|x\|_2 / \sqrt{n} \leq 4\sqrt{n}.$$

Therefore, by [5, Theorem 3.3], the MWU algorithm makes $O(\ell\rho(\log m)/\varepsilon^2) = \tilde{O}(1)$ calls to the ORACLE in Proposition 4.1, with an additional processing time of $O(m)$ per call, and provided that all the calls to the oracle succeed, either finds $x \in [-1, 1]^n$ solving the feasibility program to within an additive error of ε on the right hand side for each of the three inequalities, or correctly concludes that the system is infeasible. Since each call to the oracle succeeds with probability at least $1 - 1/\sqrt{n}$, it follows by a union bound that all calls succeed with probability at least $1 - \tilde{O}(1)/\sqrt{n}$. Moreover, the total running time is $\tilde{O}(m) + \tilde{O}(kn + n^2) = \tilde{O}(kn + n^2)$, where we have used that $m \leq n^2$. \square

In the next section, we will show how to implement the algorithm Regularized-Solve. Our construction crucially relies on the following reduction to solving a minimax program, whose solutions can be transformed into low width solutions for the MWU algorithm by using the structure of our linear program.

Proposition 4.2. *There exists a randomized algorithm Optimize for which the following holds. On input*

- a matrix $A \in \mathbb{R}^{m \times n}$ with $m \leq n^2$, $\|A\|_{1 \rightarrow \infty} \leq 1$, and $\max_{i \in [n]} |\text{supp}(Ae_i)| \leq k$,
- $\rho_0 \in \mathbb{R}^{\geq 0}$, $\rho_+, \rho_- \in (\mathbb{R}^{\geq 0})^m$ with $\rho_0 + \|\rho_+\|_1 + \|\rho_-\|_1 = 1$,
- $v \in \mathbb{R}^n$ with $\|v\|_2 \in [0, 2\sqrt{n}]$,
- $\delta \in (0, 1)$,

Optimize($A, \rho_0, \rho_+, \rho_-, v, \delta$) runs in time $\tilde{O}(kn + n^2)$, and with probability at least $1 - 1/n$, outputs a point $x' \in [-1, 1]^n$ such that

$$\begin{aligned} & -\rho_0 v^T x' / \sqrt{n} + \rho_+^T A x' - \rho_-^T A x' + \delta \|A x'\|_\infty \\ & \leq \min_{x \in [-1, 1]^n} -\rho_0 v^T x / \sqrt{n} + \rho_+^T A x - \rho_-^T A x + \delta \|A x\|_\infty + \sqrt{n}/(\log n)^4. \end{aligned}$$

We conclude this section by showing how to transform x' into a point satisfying the conclusion of Proposition 4.1.

⁶The value of Λ is not known to us. However, by combining the discussion here with a standard binary search routine, we can approximate Λ to within additive error $\sqrt{n}/(\log n)^4$ with probability $1 - o_n(1)$. The binary search procedure only blows up the overall running time by a factor of $O(\log \log n)$, since $\Lambda \in [0, 2\sqrt{n}]$ by (2.1).

Proof of Proposition 4.1 given Proposition 4.2. Let $\delta = 1/(\log n)^4$ and $C_{m,n} := C\sqrt{n \log(m/n + 2)}$. For $\rho = (\rho_0, \rho_+, \rho_-)$, define

$$\text{OPT}_\rho := \min_{x \in [-1, 1]^n} -\rho_0 v^T x / \sqrt{n} + \rho_+^T A x - \rho_-^T A x + \delta \|A x\|_\infty. \quad (4.1)$$

Note that $\text{OPT}_\rho \leq 0$, since $0 \in [-1, 1]^n$. Under the assumption that the linear program

$$\sup_{x \in \Gamma_{A,C}} \langle x, v \rangle \geq \Lambda$$

is feasible, we have that

$$\begin{aligned} \text{OPT}_\rho &\leq -\rho_0 \Lambda + (\|\rho_+\|_1 + \|\rho_-\|_1) C_{m,n} + \delta C_{m,n} \\ &\leq -\rho_0 \Lambda + (\|\rho_+\|_1 + \|\rho_-\|_1) C_{m,n} + 2C\sqrt{n}/(\log n)^{7/2}, \end{aligned} \quad (4.2)$$

where we have used the value of δ and the assumption $m \leq n^2$.

Moreover, it follows from Proposition 4.2 that in time $\tilde{O}(nk + n^2)$ and with probability at least $1 - 1/n$, we can produce a vector $x \in [-1, 1]^n$ such that

$$-\rho_0 v^T x / \sqrt{n} + \rho_+^T A x - \rho_-^T A x + \delta \|A x\|_\infty \leq \text{OPT}_\rho + \sqrt{n}/(\log n)^4.$$

To finish the proof, we show how to convert this x into a satisfying assignment for the second bullet point of Proposition 2.4 (if the linear program is feasible) or to certify infeasibility otherwise. Let $\tau := \delta \|A x\|_\infty$, which can be computed in time $O(\text{nnz}(A) + \max(m, n))$. We have the following cases:

Case I: $\tau \geq 10\sqrt{n}$. We have

$$-\rho_0 v^T x / \sqrt{n} + \rho_+^T A x - \rho_-^T A x \leq \text{OPT}_\rho + \sqrt{n}/(\log n)^4 - \tau \leq \sqrt{n} - \tau.$$

Now $y := x \cdot 10\sqrt{n}/\tau$ satisfies the conclusion of Proposition 4.1 since $y \in [-1, 1]^n$, $\|A y\|_\infty \leq 10\sqrt{n}\delta^{-1} = 10\sqrt{n}(\log n)^4$, and

$$\begin{aligned} -\rho_0 v^T y / \sqrt{n} + \rho_+^T A y - \rho_-^T A y &\leq (\sqrt{n} - \tau) \cdot 10\sqrt{n}/\tau \leq -5\sqrt{n} \\ &\leq -\Lambda \leq -\rho_0 \Lambda \\ &\leq -\rho_0 \Lambda + C_{m,n}(\|\rho_+\|_1 + \|\rho_-\|_1). \end{aligned}$$

Case II: $\tau < 10\sqrt{n}$. Then, $\|A x\|_\infty \leq 10\sqrt{n}(\log n)^4$ and

$$-\rho_0 v^T x / \sqrt{n} + \rho_+^T A x - \rho_-^T A x \leq \text{OPT}_\rho + \sqrt{n}/(\log n)^4.$$

We compute the left hand side of the above inequality in time $O(\text{nnz}(A) + m)$. If it is at most

$$-\rho_0 \Lambda + (\|\rho_+\|_1 + \|\rho_-\|_1) C_{m,n} + 2C\sqrt{n}/(\log n)^{7/2} + \sqrt{n}/(\log n)^4,$$

then x satisfies the conclusion of Proposition 4.1. If not, then it must be the case that

$$\text{OPT}_\rho > -\rho_0 \Lambda + (\|\rho_+\|_1 + \|\rho_-\|_1) C_{m,n} + 2C\sqrt{n}/(\log n)^{7/2},$$

which certifies by (4.2) that the original linear program is not feasible. \square

5. SOLVING THE MINIMAX PROGRAM VIA SUBLINEAR PRIMAL-DUAL ALGORITHM

Finally, we prove Proposition 4.2 in the following more general form.

Proposition 5.1. *There is a randomized algorithm Optimize for which the following holds. On input*

- $v_1, \dots, v_m \in \mathbb{R}^n$ with $\|v_i\|_2 \leq 1/2$ and $\max_{i \in [n]} \#\{j : \langle v_j, e_i \rangle \neq 0\} \leq k$,
- $v' \in \mathbb{R}^n$ with $\|v'\|_2 \leq 1/2$,
- $\varepsilon \in (0, 1)$,

Optimize($v', v_1, \dots, v_m, \varepsilon$) runs in time $\tilde{O}((n+k)\varepsilon^{-2} + m)$ ⁷, and with probability at least $1 - 1/n$, outputs a point $x' \in [-1/\sqrt{n}, 1/\sqrt{n}]^n$ such that

$$\max_{j \in [m]} (v' + v_j)^T x' \leq \varepsilon + \min_{x \in [-1/\sqrt{n}, 1/\sqrt{n}]^n} \max_{j \in [m]} (v' + v_j)^T x.$$

We claim that [Proposition 5.1](#) implies [Proposition 4.2](#). This is due to the following set of observations. To disambiguate notation, let $A \in \mathbb{R}^{m' \times n}$ be the matrix appearing in [Proposition 4.2](#).

- We set $m = 2m'$. For $i \in [m']$, we let $v_i := \delta \cdot e_i^T A / (2\sqrt{n})$ and $v_{m'+i} = -\delta \cdot e_i^T A / (2\sqrt{n})$. The assumptions on v_1, \dots, v_m are satisfied by the first bullet point of [Proposition 4.2](#) and since $\delta \in (0, 1)$. Moreover, for any $x \in \mathbb{R}^n$,

$$\frac{\delta}{2\sqrt{n}} \|Ax\|_\infty = \max_{i \in [m]} v_i^T x.$$

- We let $v' := (-\rho_0 v^T / \sqrt{n} + \rho_+^T A - \rho_-^T A) / 2\sqrt{n}$. Note that this vector can be computed in time $O(\text{nnz}(A) + \max\{m, n\}) = O(\text{nnz}(A) + n^2) = O(nk + n^2)$. The norm assumption on v' holds since

$$\|-\rho_0 v^T / \sqrt{n} + \rho_+^T A - \rho_-^T A\|_2 \leq \max\{\|v\|_2 / \sqrt{n}, \|A\|_{2 \rightarrow \infty}\} \leq \sqrt{n},$$

where the final inequality uses the first and third bullet points of [Proposition 4.1](#).

- Finally, setting $\varepsilon = 1/(2\sqrt{n}(\log n)^4)$, it is immediately seen that if x' satisfies the conclusion of [Proposition 5.1](#), then $x = 2\sqrt{n} \cdot x'$ satisfies the conclusion of [Proposition 4.2](#).

We will prove [Proposition 5.1](#) using the sublinear primal-dual framework of Clarkson, Hazan, and Woodruff [[12](#), Algorithm 1, Algorithm 3]. The pseudocode is presented in [Algorithm 1](#) and relies on a few subroutines, which we now discuss.

First, we need a standard iterative low-regret algorithm for the class of ‘experts’ corresponding to the rescaled continuous cube $\mathcal{C} = [-1/\sqrt{n}, 1/\sqrt{n}]^n$.

Lemma 5.2. Consider a sequence of vectors $v_\ell \in \mathbb{R}^n$ with $\|v_\ell\|_2 \leq 1$ for $\ell \in [T]$ and let $\mathcal{C} = [-1/\sqrt{n}, 1/\sqrt{n}]^n$. The sequence of vectors $x_0 = 0$ and for $i \geq 1$,

$$x_{i+1} = \text{LRA}(v_i, x_{i-1}) := \arg \min_{x \in \mathcal{C}} \|x - (x_{i-1} - \eta v_i)\|_2^2,$$

where $\eta = \sqrt{2/T}$, satisfies

$$\sup_{\ell \in [T]} \frac{1}{T} \left(\max_{x \in \mathcal{C}} \sum_{i=1}^{\ell} v_i^T x - \sum_{i=1}^{\ell} v_i^T x_i \right) \leq \sqrt{\frac{2}{T}}.$$

Moreover, given v_i and x_{i-1} , $\text{LRA}(v_i, x_{i-1})$ can be computed in time $O(n)$.

Proof. The expression for $\text{LRA}(\cdot, \cdot)$ corresponds exactly to online mirror descent on \mathcal{C} equipped with the ℓ_2 -norm, with respect to the 1-strongly convex mirror map $\Phi(x) = \frac{\|x\|_2^2}{2}$. Accordingly, we have the standard guarantee (see, e.g., [[10](#), Equation 4.10]) that for any $x \in \mathcal{C}$,

$$\frac{1}{T} \sum_{i=1}^{\ell} v_i^T (x - x_i) \leq \frac{1}{T} \left(\sup_{z \in \mathcal{C}} \frac{\|z\|_2^2}{\eta} + \frac{\eta}{2} \sum_{i=1}^{\ell} \|v_i\|_2^2 \right) \leq \frac{1}{T\eta} + \frac{\eta}{2} = \sqrt{\frac{2}{T}},$$

taking the balancing value $\eta = \sqrt{2/T}$. For the assertion about the running time, note that $(x_{i-1} - \eta v_i)$ can be readily computed in time $O(n)$, and the minimization to compute x_{i+1} can also be performed in time $O(n)$, since the closest point in \mathcal{C} to a given point in \mathbb{R}^n can be found in a coordinate-by-coordinate manner. \square

⁷We make the standard data-structure assumption that for any $i \in [n]$, the set $\{j : \langle v_j, e_i \rangle \neq 0\}$, which has size at most k by assumption, can be determined in time $\tilde{O}(k)$.

Next, for any $v \in \mathbb{R}^n$ with $\|v\|_2 \leq 1$ and $x \in \mathcal{C}$, we need a fast, low-variance, unbiased estimator for $v^T x$. This is the content of the following lemma.

Lemma 5.3. *Let $x, v \in \mathbb{R}^n$ with $\|x\|_2, \|v\|_2 \leq 1$. Define $\text{Est}(x, v)$ to be v_i/x_i with probability equal to x_i^2 and 0 with probability $1 - \|x\|_2^2$. Then, we have that*

$$\mathbb{E}[\text{Est}(x, v)] = x^T v \quad \text{and} \quad \text{Var}[\text{Est}(x, v)] \leq 1.$$

Proof. Note that

$$\mathbb{E}[\text{Est}(x, v)] = \sum_{i \in [n]} v_i/x_i \cdot x_i^2 = v^T x$$

and

$$\text{Var}[\text{Est}(x, v)] \leq \mathbb{E}[\text{Est}(x, v)^2] = \sum_{i \in [n]} v_i^2/x_i^2 \cdot x_i^2 \leq 1. \quad \square$$

Finally, we require a data-structure which supports efficiently updating and sampling from probability distributions. We use (a much simpler version of) a data-structure provided in work of Carmon, Jin, Sidford, and Tian [11, Section 2.4.1].

Lemma 5.4. *There exists a data-structure for handling probability distributions on $[n]$ with the follow properties:*

- *Initialization(v):* Given $v \in (\mathbb{R}_{>0})^n$, construct the data structure corresponding to the probability distribution $v/\|v\|_1$ on $[n]$ in time $O(n)$.
- *Mult(v, i, τ):* Given the data structure corresponding to the probability distribution determined by $v \in (\mathbb{R}_{\geq 0})^n$, a coordinate $i \in [n]$, and $\tau \in \mathbb{R}_{\geq 0}$, update to the data structure corresponding to the probability distribution determined by $v' \in (\mathbb{R}_{\geq 0})^n$ in time $O(\log n)$, where $v'_i = \tau v_i$ and $v'_j = v_j$ for $j \neq i$.
- *Sample(v):* Given the data structure corresponding to the probability distribution determined by $v \in (\mathbb{R}_{\geq 0})^n$, produce a sample according to it in time $O(\log n)$.

Proof sketch. The data structure associated to $v \in (\mathbb{R}_{\geq 0})^n$ consists of an array on $[n]$, storing the entries of v , with a full binary tree on top. Each node in the tree maintains the sum of all the elements in the array which are its descendants (in particular, the sum at the root is $\|v\|_1$). Initialize(v) constructs this tree in a ‘bottom-to-top’ fashion and takes time $O(n)$ since there are $O(n)$ edges in this tree; Mult(v, i, τ) is implemented by starting at the i^{th} position and ‘walking-up’ to the root along the unique root-to-leaf path, updating the weights of the $O(\log n)$ nodes encountered on the way; Sample(v) is implemented in $O(\log n)$ -time by ‘walking down’ from the root to a leaf, using the values at the left and right child of each node in order to toss a coin with suitable bias and decide whether to descend to the left child or to the right child. \square

The pseudocode for Proposition 5.1 is given in Algorithm 1. For a given vector $x \in \mathbb{R}^n$ with $\|x\|_2 \leq 1$, we define $\text{Dist}(x, \ell_2)$ to be the distribution $[n]$ specified by the square of the coordinates (outputting \emptyset with probability $1 - \|x\|_2^2$) and for $z, C \in \mathbb{R}$, define $\text{clip}(z, C) = \min\{\max\{z, -C\}, C\}$. From the above description of various subroutines, it is immediate that OPTIMIZE runs in the required time.

Lemma 5.5. *The runtime for OPTIMIZE($v', v_1, \dots, v_m, \varepsilon$) is bounded by $\tilde{O}((k+n)\varepsilon^{-2} + m)$.*

Proof. Lines 1-4 take time $O(n+m)$. Line 15 takes time $O(Tn) = \tilde{O}(\varepsilon^{-2}n)$. By Lemma 5.4, each iteration of Lines 6-7 can be implemented in time $\tilde{O}(n)$, for a total runtime of $O(nT) = \tilde{O}(\varepsilon^{-2}n)$. By Lemma 5.2, each iteration of Line 14 takes time $O(n)$, for a total runtime of $O(nT) = \tilde{O}(\varepsilon^{-2}n)$. Each iteration of Line 9 takes time $O(1)$ and each iteration of Line 10 takes time $\tilde{O}(k)$ (see the footnote in the statement of Proposition 5.1), so that together, all iterations of Lines 9-10 take time

Algorithm 1: Pseudocode for OPTIMIZE($v', v_1, \dots, v_m, \varepsilon$) in Proposition 5.1

```

1  $T \leftarrow (\log m)\varepsilon^{-2}$ 
2  $x_0 \leftarrow 0$ 
3  $w_0 \leftarrow \text{Initialize}(\mathbb{1}_m)$ 
4  $\eta \leftarrow \sqrt{(\log m)/T}/100$ 
5 for  $t = 1, \dots, T$  do
6    $\tau_t \leftarrow \text{Sample}(\text{Dist}(x_{t-1}, \ell_2))$ 
7    $s_t \leftarrow \text{Sample}(w_{t-1})$ 
8   if  $\tau_t \neq \emptyset$  then
9      $v_t^* \leftarrow \text{Clip}(v^T e_{\tau_t}/x_{\tau_t}, 1/\eta)$ 
10     $J_t \leftarrow \{j \in [m] : v_j^T e_{\tau_t} \neq 0\}$ 
11    for  $j \in J_t$  do
12       $v_t(j) \leftarrow \text{Clip}((v' + v_j)^T e_{\tau_t}/x_{\tau_t}, 1/\eta)$ 
13       $w_t(j) \leftarrow \text{Mult}(w_{t-1}, j, (1 - \eta v_t(j) + \eta^2 v_t(j)^2) \cdot (1 - \eta v_t^* + \eta^2 (v_t^*)^2)^{-1})$ 
14     $x_t \leftarrow \text{LRA}(x_{t-1}, v' + v_{s_t})$ 
15 return  $\frac{1}{T} \sum_{i=1}^T x_i$ 

```

$\tilde{O}(Tk) = \tilde{O}(k\varepsilon^{-2})$. Finally, by Lemma 5.4, each iteration of Lines 11-13 takes time $\tilde{O}(k)$, for a total runtime of $\tilde{O}(kT) = \tilde{O}(k\varepsilon^{-2})$. \square

Finally, we analyse the correctness of Algorithm 1.

Proof of Proposition 5.1. The running time of OPTIMIZE is analyzed in Lemma 5.5. For the correctness, it is helpful to note that the probability distribution defined by the vector $w_t \in (\mathbb{R}_{\geq 0})^m$ in Line 13 coincides with the probability distribution defined by the vector $w'_t \in (\mathbb{R}_{\geq 0})^m$, where for $j \in [m]$,

$$w'_t(j) := w_{t-1}(j) \cdot (1 - \eta v_t(j) + \eta^2 v_t(j)^2),$$

for $v_t(j)$ defined by the same formula as Line 12 (but now, for all $j \in [m]$). Indeed, $w_t = (1 - \eta v_t^* + \eta^2 (v_t^*)^2)^{-1} \cdot w'_t$.

With this observation, the conclusion follows essentially immediately from [12, Algorithm 3], upon noting that

- $T_\varepsilon(\text{LRA})$ is any T for which the right hand side of Lemma 5.2 is bounded above by ε ; clearly, $T \geq 4/\varepsilon^2$ suffices, and
- by Lemma 5.3 and since $\|v' + v_j\|_2 \leq 1$ for all $j \in [m]$, we have that $(v' + v_j)^T e_{\tau_t}/x_{\tau_t}$ is an unbiased estimator for $(v' + v_j)^T x$ with variance bounded by 1.

The only difference between [12, Algorithm 3] and Algorithm 1 is that in Algorithm 1, the estimators $(v' + v_j)^T e_{\tau_t}/x_{\tau_t}$ are not independent for different $j \in [m]$. This only affects (potentially) the proofs of [12, Lemma B.3, Lemma B.6, Lemma B.7] but a trivial examination of the proof reveals that independence between the estimators is not used ⁸. \square

Remark (Numerical Precision Issues). The above analysis, as written, assumes exact arithmetic; there are two points which are numerically sensitive which can be handled using standard techniques.

⁸This observation is already present in [12, Algorithm 1]. In fact, our setting is essentially identical to [12, Algorithm 1], except that there, the minimization is over x with $\|x\|_2 \leq 1$. Except for the use of the data structure in Line 13, our algorithm is identical to [12, Algorithm 1] modulo noting that $[-1/\sqrt{n}, 1/\sqrt{n}]^n$ is contained in the unit ℓ_2 -ball and that the associated projection in mirror descent can be implemented efficiently.

The first is dividing by x_t in Lines 9 and 11. By [12, Lemma C.2], it suffices to truncate entries smaller than $\text{poly}(n^{-1}, \varepsilon)$.

The second point is keeping track of the vector w_t , and the induced probability distribution, in a numerically stable manner. This is discussed [11, Section G.1]; for our simplified data structure, however, a substantially simpler solution suffices, which we now sketch.

First, note that the maximum and minimum value of any $w_t(i)$ over the course of the algorithm is bounded between 4^{-T} and 4^T . We will maintain the logarithm of each $w_t(i)$ using $L = C(\log n + \log(1/\varepsilon))$ bits, for a sufficiently large constant C . In particular, the version of $w_t(i)$ we work with is within a factor of $(1 \pm \varepsilon^C n^{-C})^{\tilde{O}((k+n)\varepsilon^{-2+m})}$ of the true $w_t(i)$, which is negligible for C sufficiently large.

For maintaining the logarithm of the weights up to this precision in Line 13, note that, when ‘walking up’ the binary tree, it suffices to set the logarithm of the value of a parent node to be equal to the logarithm of the value of the heavier child node, in the case when the logarithms of the values of the children differ by more than $2L$. Otherwise, denoting the value of the lighter child by z and the heavier child by y , the logarithm of the value at the parent node is $\log(y+z) = \log(y) + \log(1+z/y)$, which can be computed in $O(\text{poly}(\log n, \log(1/\varepsilon)))$ -time to L digits of precision, since $2^{-2L} \leq |z/y| \leq 1$. Finally, when ‘walking down’ the binary tree in Line 7, if the logarithms of the values of the children of a node differ by more than L , then it suffices to simply descend to the heavier child. Otherwise, the bias of the coin to flip is $y/(y+z) = 1/(1+z/y)$, which can be computed to the desired accuracy in time $O(\text{poly}(\log n, \log(1/\varepsilon)))$, noting again that $2^{-L} \leq |z/y| \leq 1$.

REFERENCES

- [1] Zeyuan Allen-Zhu and Lorenzo Orecchia, *Nearly linear-time packing and covering LP solvers*, Mathematical Programming **175** (2019), 307–353. [4](#)
- [2] Josh Alman and Virginia Vassilevska Williams, *A refined laser method and faster matrix multiplication*, Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), [Society for Industrial and Applied Mathematics (SIAM)], Philadelphia, PA, 2021, pp. 522–539. [1](#)
- [3] Noga Alon and Joel H. Spencer, *The probabilistic method*, fourth ed., Wiley Series in Discrete Mathematics and Optimization, John Wiley & Sons, Inc., Hoboken, NJ, 2016. [1](#)
- [4] Ryan Alweiss, Yang P. Liu, and Mehtaab Sawhney, *Discrepancy minimization via a self-balancing walk*, STOC ’21—Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, ACM, New York, [2021] ©2021, pp. 14–20. [2](#), [5](#), [17](#)
- [5] Sanjeev Arora, Elad Hazan, and Satyen Kale, *The multiplicative weights update method: a meta-algorithm and applications*, Theory of computing **8** (2012), 121–164. [3](#), [10](#), [11](#)
- [6] Wojciech Banaszczyk, *Balancing vectors and Gaussian measures of n -dimensional convex bodies*, Random Structures & Algorithms **12** (1998), 351–360. [2](#)
- [7] Nikhil Bansal, *Constructive algorithms for discrepancy minimization*, 2010 IEEE 51st Annual Symposium on Foundations of Computer Science—FOCS 2010, IEEE Computer Soc., Los Alamitos, CA, 2010, pp. 3–10. [1](#)
- [8] József Beck, *Roth’s estimate of the discrepancy of integer sequences is nearly sharp*, Combinatorica **1** (1981), 319–325. [1](#)
- [9] József Beck and Tibor Fiala, *“integer-making” theorems*, Discrete Applied Mathematics **3** (1981), 1–8. [2](#)
- [10] Sébastien Bubeck, *Convex optimization: Algorithms and complexity*, Foundations and Trends® in Machine Learning **8** (2015), 231–357. [13](#)
- [11] Yair Carmon, Yujia Jin, Aaron Sidford, and Kevin Tian, *Coordinate methods for matrix games*, 2020 IEEE 61st Annual Symposium on Foundations of Computer Science, IEEE Computer Soc., Los Alamitos, CA, [2020] ©2020, pp. 283–293. [14](#), [16](#)
- [12] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff, *Sublinear optimization for machine learning*, J. ACM **59** (2012), Art. 23, 49. [5](#), [13](#), [15](#), [16](#)
- [13] Daniel Dadush, <https://homepages.cwi.nl/~dadush/workshop/discrepancy-ip/open-problems.html>. [2](#)
- [14] Ronen Eldan and Mohit Singh, *Efficient algorithms for discrepancy minimization in convex sets*, Random Structures Algorithms **53** (2018), 289–307. [1](#), [2](#), [3](#), [7](#), [8](#)
- [15] Apostolos A. Giannopoulos, *On some vector balancing problems*, Studia Math. **122** (1997), 225–234. [1](#)

- [16] E. D. Gluskin, *Extremal properties of orthogonal parallelepipeds and their applications to the geometry of Banach spaces*, Mat. Sb. (N.S.) **136(178)** (1988), 85–96. [1](#)
- [17] Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang, *Faster dynamic matrix inverse for faster lps*. [1](#)
- [18] Avi Levy, Harishchandra Ramadas, and Thomas Rothvoss, *Deterministic discrepancy minimization via the multiplicative weight update method*, Integer programming and combinatorial optimization, Lecture Notes in Comput. Sci., vol. 10328, Springer, Cham, 2017, pp. 380–391. [1](#)
- [19] Yang P. Liu, Ashwin Sah, and Mehtaab Sawhney, *A Gaussian Fixed Point Random Walk*, 13th Innovations in Theoretical Computer Science Conference (ITCS 2022) (Dagstuhl, Germany) (Mark Braverman, ed.), Leibniz International Proceedings in Informatics (LIPIcs), vol. 215, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 101:1–101:10. [2](#), [5](#)
- [20] Shachar Lovett and Raghu Meka, *Constructive discrepancy minimization by walking on the edges*, SIAM J. Comput. **44** (2015), 1573–1582. [1](#)
- [21] Thomas Rothvoss, *Constructive discrepancy minimization for convex sets*, SIAM J. Comput. **46** (2017), 224–234. [1](#), [2](#), [3](#), [6](#)
- [22] Joel Spencer, *Six standard deviations suffice*, Trans. Amer. Math. Soc. **289** (1985), 679–706. [1](#), [3](#), [7](#), [8](#)

APPENDIX A. DEFERRED PROOFS FROM SECTION 2

A.1. **Proof of Theorem 1.1 given Theorem 2.1.** The following result is an immediate consequence of [4, Theorem 1.2].

Theorem A.1. *There is a randomized algorithm Sparse-Coloring and an absolute constant $C_{A.1} > 0$ for which the following holds. On input a matrix $A \in \mathbb{R}^{m \times n}$, Sparse-Coloring(A) runs in time $O(\text{nnz}(A) + n)$, and with probability at least 99/100, returns a vector $v \in \{\pm 1\}^n$ such that*

$$\|Av\|_\infty \leq C_{A.1} \|A\|_{1 \rightarrow 2} \sqrt{(\log m)(\log n)}.$$

Proof of Theorem 1.1. First, note that we may assume that $m \leq n^2$; otherwise Theorem 1.1 follows from noting (using Bernstein’s inequality and the union bound) that a uniformly random $v \in \{\pm 1\}^n$ succeeds with probability at least 1/2 (and given v , its success can be checked in time $O(\text{nnz}(A) + n)$). Moreover, we may assume that $m \geq n/(\log n)^2$; otherwise, $\|A\|_{1 \rightarrow 2} \leq \sqrt{m} \cdot \|A\|_{1 \rightarrow \infty} \leq \sqrt{n}/(\log n)$, and we may use the algorithm Sparse-Coloring from Theorem A.1.

Now, suppose that $n/(\log n)^2 \leq m \leq n^2$. Define the sets

$$C_{\text{Light}} := \{i \in [n] : \|Ae_i\|_2 \leq \sqrt{n}/(\log n)\} \quad \text{and} \quad C_{\text{Heavy}} := [n] \setminus C_{\text{Light}},$$

where e_i denote the i -th elementary basis vector. Note that for any $j \in C_{\text{Heavy}}$, we have

$$\sqrt{\text{nnz}(Ae_j)} \|A\|_{1 \rightarrow \infty} \geq \sqrt{n}/(\log n),$$

from which we see that

$$\text{nnz}(Ae_j) = \tilde{\Omega}(n).$$

We define A_{Light} to be the restriction of A to columns corresponding to $\mathbb{R}^{C_{\text{Light}}}$ and A_{Heavy} to be the restriction of A to columns corresponding to $\mathbb{R}^{C_{\text{Heavy}}}$. Note that it suffices to find a vector $v \in \{\pm 1\}^{C_{\text{Heavy}}}$ such that $\|A_{\text{Heavy}}v\|_\infty \lesssim \sqrt{n \log(m/n + 2)}$ in time $\tilde{O}(\text{nnz}(A) + n)$ as A_{Light} is handled immediately by Theorem A.1. By Theorem 2.1, we can find such a v in time $\tilde{O}(\text{nnz}(A_{\text{Heavy}}) + |C_{\text{Heavy}}|^2) = \tilde{O}(\text{nnz}(A) + |C_{\text{Heavy}}|^2)$, which we claim is $\tilde{O}(\text{nnz}(A) + n)$. Indeed,

$$\text{nnz}(A_{\text{Heavy}}) \geq |C_{\text{Heavy}}| \cdot \min_{j \in C_{\text{Heavy}}} \text{nnz}(Ae_j) = \tilde{\Omega}(|C_{\text{Heavy}}| \cdot n)$$

so that

$$|C_{\text{Heavy}}|^2 \leq |C_{\text{Heavy}}| \cdot n = \tilde{O}(\text{nnz}(A_{\text{Heavy}})) = \tilde{O}(\text{nnz}(A)). \quad \square$$

A.2. Proof of Theorem 2.1 given Theorem 2.2.

Proof of Theorem 2.1. As in the previous subsection, it suffices to consider the case $n/(\log n)^2 \leq m \leq n^2$.

Initialize $v_0 = 0$. At each time step $\ell \geq 1$, given the partial coloring $v_{\ell-1} \in [-1, 1]^n$, let $\mathcal{F}_\ell = \{i \in [n] : (v_{\ell-1})_i \in \{\pm 1\}\}$, $\mathcal{G}_\ell = \{i \in [n] : (v_{\ell-1})_i \notin \{\pm 1\}\}$, and $\Lambda_\ell = \text{Diag}(1 - |v_{\ell-1}|)$. If $\mathcal{G}_\ell = \emptyset$, then return $v_{\ell-1}$. Else, let A_ℓ denote the restriction of A to the columns spanned by \mathcal{G}_ℓ and notice that Λ_ℓ restricts naturally to \mathcal{G}_ℓ .

We consider two separate cases. If $|\mathcal{G}_\ell| \geq n/(\log n)^2$, then we use Theorem 2.2 to find a vector $v'_\ell \in [-1, 1]^{|\mathcal{G}_\ell|}$ such that $\|A_\ell \Lambda_\ell v'_\ell\|_\infty \leq C_{2.2} \sqrt{|\mathcal{G}_\ell| \log(m/|\mathcal{G}_\ell| + 2)}$ and such that v'_ℓ has at least $C_{2.2}^{-1} |\mathcal{G}_\ell|$ coordinates that are valued in $\{\pm 1\}$ ⁹. In particular, for at least one value of $\sigma_\ell \in \{\pm 1\}$, the vector $v_\ell := v_{\ell-1} + \sigma_\ell \Lambda_\ell v'_\ell$ has at most $(1 - C_{2.2}^{-1}/2) |\mathcal{G}_\ell|$ coordinates that are not valued in $\{\pm 1\}$. Moreover,

$$\|Av_\ell\|_\infty \leq \|Av_{\ell-1}\|_\infty + \|A_\ell \Lambda_\ell v'_\ell\|_\infty \leq \|Av_{\ell-1}\|_\infty + C_{2.2} \sqrt{|\mathcal{G}_\ell| \log(m/|\mathcal{G}_\ell| + 2)}.$$

On the other hand, if $|\mathcal{G}_\ell| \leq n/(\log n)^2$, let $v'_\ell \in [-1, 1]^n$ denote the random vector with independent coordinates such that $(v'_\ell + v_{\ell-1})_i \in \{\pm 1\}$ and $\mathbb{E}[(v'_\ell)_i] = 0$ for all $i \in [n]$. Let $v_\ell = v'_\ell + v_{\ell-1} \in \{\pm 1\}^n$. A direct application of Bernstein's inequality and the union bound shows that, with probability at least 99/100,

$$\|Av_\ell\|_\infty \leq \|Av_{\ell-1}\|_\infty + \|Av'_\ell\|_\infty \leq \|Av_{\ell-1}\|_\infty + 10\sqrt{n}.$$

Note that, given $v_{\ell-1}$, we can sample from v'_ℓ in time $O(n)$ and verify that v_ℓ satisfies the above inequality in time $O(\text{nnz}(A) + n)$.

Observe that, due to the guarantee $|\mathcal{G}_{\ell+1}| \leq (1 - C_{2.2}^{-1}/2) |\mathcal{G}_\ell|$, we are in the first case for at most $O(\log \log n)$ iterations, which together take time

$$\tilde{O} \left(\sum_{\ell=0}^{O(\log \log n)} (1 - C_{2.2}^{-1}/2)^{2\ell-2} n^2 + \text{nnz}(A_\ell) \right) = \tilde{O}(n^2 + \text{nnz}(A)).$$

As mentioned before, the second step takes time $O(\text{nnz}(A) + n)$.

Finally, denoting the output of the process by $v \in \{\pm 1\}^n$ and using $|\mathcal{G}_\ell| \lesssim (1 - C_{2.2}^{-1}/2)^\ell n$, we have that

$$\|Av\|_\infty \lesssim \sqrt{n} + \sum_{\ell \leq O(\log \log n)} \sqrt{|\mathcal{G}_\ell| \log(m/|\mathcal{G}_\ell| + 2)} \lesssim \sqrt{n \log(m/n + 2)},$$

as desired. □

DEPARTMENT OF STATISTICS, STANFORD UNIVERSITY, STANFORD, CA 94305, USA
Email address: visheshj@stanford.edu

DEPARTMENT OF MATHEMATICS, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MA 02139, USA
Email address: {asah, msawhney}@mit.edu

⁹More precisely, we make at most $\tilde{O}(1)$ independent calls to Theorem 2.2, which guarantees that we find such a v'_ℓ with probability at least $1 - 1/n$.