

Acceleration Meets Inverse Maintenance: Faster ℓ_∞ -Regression

Deeksha Adil
Institute for Theoretical Studies
ETH Zürich
deeksha.adil@eth-its.ethz.ch

Shunhua Jiang
Department of Computer Science
Columbia University
sj3005@columbia.edu

Rasmus Kyng
Department of Computer Science
ETH Zürich
kyng@inf.ethz.ch

Abstract

We propose a randomized multiplicative weight update (MWU) algorithm for ℓ_∞ regression that runs in $\tilde{O}\left(n^{2+1/22.5}\text{poly}(1/\epsilon)\right)$ time when $\omega = 2 + o(1)$, improving upon the previous best $\tilde{O}\left(n^{2+1/18}\text{poly}\log(1/\epsilon)\right)$ runtime in the low-accuracy regime. Our algorithm combines state-of-the-art inverse maintenance data structures with acceleration. In order to do so, we propose a novel acceleration scheme for MWU that exhibits *stability* and *robustness*, which are required for the efficient implementations of the inverse maintenance data structures.

We also design a faster *deterministic* MWU algorithm that runs in $\tilde{O}\left(n^{2+1/12}\text{poly}(1/\epsilon)\right)$ time when $\omega = 2 + o(1)$, improving upon the previous best $\tilde{O}\left(n^{2+1/6}\text{poly}\log(1/\epsilon)\right)$ runtime in the low-accuracy regime. We achieve this by showing a novel stability result that goes beyond previously known works based on interior point methods (IPMs).

Our work is the first to use acceleration and inverse maintenance together efficiently, finally making the two most important building blocks of modern structured convex optimization compatible.

Contents

1	Introduction	4
1.1	Our Results	4
1.2	Background: The Ingredients of Fast ℓ_∞ -Regression Methods.	5
1.3	Discussion of Techniques	6
2	Technical Overview	9
2.1	Deterministic MWU Algorithm via One-Level Inverse Maintenance	9
2.2	Randomized MWU Algorithm via Two-Level Inverse Maintenance	10
3	Fast Width-Reduced MWU Algorithms	13
3.1	Lazy Update Procedure	13
3.2	Monotone Multiplicative Weights Update Algorithm	14
3.3	Algorithm with Non-Monotone Weights, Stability and Robustness	14
A	Preliminaries	20
B	Low Rank Update Scheme under Stability Guarantees	21
B.1	Low Rank Update under ℓ_2 Stability	21
B.2	Low Rank Update under ℓ_3 Stability	22
B.2.1	Decomposition of Iterations	22
B.2.2	Low Rank Update Scheme under ℓ_3 Stability	23
C	Data Structures	25
C.1	Inverse Maintenance Data Structure	25
C.2	Implicit Inverse Maintenance	26
C.3	ℓ_3 and ℓ_2 -Norm Estimations	27
C.4	ℓ_2 Heavy Hitter	28
D	Time Complexity of the Randomized Algorithm Using Fast Data Structures	28
D.1	Implementing MWU Using Fast Data Structures	28
D.2	Correctness of Algorithm	28
D.3	Time Complexity under ℓ_2 Stability	34
E	Time Complexity of the Deterministic Algorithm Using Fast Data Structures	37
F	Guarantees of Algorithm 1: MWU with Monotone Weights	39
G	Guarantees of Algorithm 3: Robust Primal Step and Stable Width Reduction Step	42
G.1	Starting Point: Non-Monotone MWU Algorithm	42
G.2	Warm Up: Stable Width Reduction Step	43
G.2.1	Definitions and Basic Properties	43
G.2.2	Change in Φ	45
G.2.3	Change in Ψ	46
G.2.4	Putting Everything Together: Analysis of Algorithm	53
G.3	Guarantees of Algorithm 3: Robust Primal Step	55
G.3.1	Sketching Bounds	55
G.3.2	Analysis of Algorithm 3	61

H	Stability Guarantees of Algorithms 1 and 3	66
H.1	Stability Guarantees of Algorithm 1	67
H.2	Algorithm Warm Up: Low-Rank Update Scheme	69
H.3	Algorithm 3	71
I	Missing Proofs	76
J	MWU with Non-Monotone Weights and $n^{1/3}$ Iterations (Optional)	83

1 Introduction

In this paper, we study the ℓ_∞ -regression problem. Given $\epsilon > 0$, a matrix $\mathbf{C} \in \mathbb{R}^{n \times d}$ and vector $\mathbf{d} \in \mathbb{R}^n$, $d \leq n$, we want to find $\tilde{\mathbf{x}} \in \mathbb{R}^d$ such that,

$$\|\mathbf{C}\tilde{\mathbf{x}} - \mathbf{d}\|_\infty \leq (1 + \epsilon) \min_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{C}\mathbf{x} - \mathbf{d}\|_\infty. \quad (1)$$

Some of the popular approaches to obtaining fast algorithms for ℓ_∞ -regression include using multiplicative weight update (MWU) routines [BN51; AHK12; Chr+11; Chi+13; Adi+19; EV19; ABS21], gradient descent [She13; Kel+14] and other ways to optimize a softmax function [Car+20; ST18; ABS21], and using interior point methods (IPM) [Kar84; Ren88; NN94]. Interior point methods can find a high-accuracy solution, i.e., an ϵ -approximate solution in $\tilde{O}(\sqrt{n} \log(1/\epsilon))^1$ linear system solves, whereas most of the other methods are low accuracy solvers, i.e., their running time scales as $\text{poly}(1/\epsilon)$. Naively using gradient descent or MWU requires $O(\sqrt{n} \cdot \text{poly}(1/\epsilon))$ linear system solves. Multiplicative weight update based approaches can be accelerated via a technique called *width reduction* to converge in $O(n^{1/3} \cdot \text{poly}(1/\epsilon))$ linear system solves [Chr+11; Chi+13; Adi+19; EV19; ABS21; Adi+24]. Several acceleration techniques have also been developed to improve the iteration complexity of other low-accuracy regression algorithms [MS13; Bul18; Car+20; ST18; ABS21].

To get an overall fast runtime, apart from improving the iteration complexity, a useful approach is to reduce the per-iteration cost. This can be done using *inverse maintenance*, which reduces the cost via *lazy-update* schemes. Notions of inverse maintenance appear in the very first interior point methods, [Kar84; NN89], but the modern form was introduced by Vaidya [Vai89]. There have been many important developments in inverse maintenance algorithms since then, and state-of-the-art algorithms use both linear algebraic data structures and dimensionality reduction routines, such as sketching [BNS19]. The improvements in runtimes of interior point methods including the state-of-the-art algorithms depend heavily on these developments in inverse maintenance routines [LS15; CLS21; Bra20; Jia+21; LV21].

1.1 Our Results

For simplicity, in the discussion of our results and prior work on this problem, we focus on the case $\omega = 2 + o(1)$ – but our full technical theorems give results for all ω . In the low-accuracy regime of $\epsilon = 1/\text{polylog}(n)$ the state-of-the-art running time for ℓ_∞ -regression is $\tilde{O}(n^{2+1/18})$, obtained via the randomized algorithm of [Jia+21], and $\tilde{O}(n^{2+1/6})$ for deterministic algorithms via [Bra20]. Both these algorithms in fact obtain high-accuracy solutions, and they use inverse maintenance, but no acceleration. In this work, we push the running time further in the low-accuracy regime by combining the state-of-the-art inverse maintenance techniques of these results with new multiplicative weight methods which allow us to perform acceleration, yielding running times of $\tilde{O}(n^{2+1/22.5} \text{poly}(\epsilon^{-1}))$ with randomization and $\tilde{O}(n^{2+1/12} \text{poly}(\epsilon^{-1}))$ without.

Our first result is a deterministic algorithm that combines acceleration and lazy inverse updates in a novel, more sophisticated way, and achieves a running time of $\tilde{O}(n^{2+1/12} \text{poly}(\epsilon^{-1}))$. This improves on deterministic state-of-the-art $\tilde{O}(n^{2+1/6} \log(\epsilon^{-1}))$ [Bra20] in the low-accuracy regime. The key to this result is a new notion of ℓ_3 -stability which is tailored to the accelerated MWU.

Theorem 1.1 (Informal statement of Theorem E.1). *There is a deterministic algorithm that solves Problem (1) in $\tilde{O}(n^{2+1/12} \text{poly}(\epsilon^{-1}))$ time when $\omega = 2 + o(1)$. This algorithm converges in $\tilde{O}(n^{1/3} \text{poly}(\epsilon^{-1}))$ iterations.*

¹We use $\tilde{O}(\cdot)$ to hide $\text{poly log } n$ factors, and we use $\tilde{O}_\epsilon(\cdot)$ to additionally hide $\text{poly}(\epsilon^{-1})$ factors.

Our main result is our randomized algorithm with running time $\tilde{O}(n^{2+1/22.5} \text{poly}(\epsilon^{-1}))$.

Theorem 1.2 (Informal statement of Theorem D.2). *There is a randomized algorithm that solves Problem (1) in $\tilde{O}(n^{2+1/22.5} \text{poly}(\epsilon^{-1}))$ time when $\omega = 2 + o(1)$. This algorithm converges in $\tilde{O}(n^{1/2.5} \text{poly}(\epsilon^{-1}))$ iterations.*

To obtain this result, we introduce the first MWU which can combine all three key techniques for ℓ_∞ -regression: (a) acceleration, (b) lazy inverse updates, and (c) sketching.

Thus, we give the optimization approach method which is able to efficiently combine these three key techniques of structured convex optimization. This is likely an essential building block toward $n^{2+o(1)}$ optimization for many objectives. If, some day, acceleration is achieved for linear programming, an equivalent integration will be necessary for optimal algorithms in this context. Before describing our new approach, we first review existing techniques for fast ℓ_∞ -regression.

1.2 Background: The Ingredients of Fast ℓ_∞ -Regression Methods.

Both MWUs and IPMs that solve ℓ_∞ -regression methods rely on a sequence of calls to ℓ_2 -oracles, i.e. a subroutine that solves an ℓ_2 -minimization problem, or equivalently, solves a linear equation. In order to solve the ℓ_∞ -regression problem (1), a standard MWU approach repeatedly solves a sequence of ℓ_2 -oracle problems of the form

$$\mathbf{x}^{(i)} = \arg \min_{\mathbf{x} \in \mathbb{R}^d} \sum_e r_e^{(i)} (\mathbf{C} \mathbf{x} - \mathbf{d})_e^2 \quad (2)$$

where the weights $\{r_e^{(i)}\}$ are chosen by the MWU depending on the magnitude of previous iterates.

Inverse maintenance via stability and robustness. The ℓ_2 -oracles of MWUs and IPMs can be implemented by applying the inverse of a matrix, and inverse maintenance can be used to solve the sequence of ℓ_2 -oracle calls faster than simply performing a full matrix inversion or linear equation solve on each call. Two key phenomena drive inverse maintenance: *stability* and *robustness*. *Stability* is the property that the inputs to the ℓ_2 -oracle only change slowly. In the MWU case, this means the weights $\{r_e^{(i)}\}$ change slowly. We say an optimizer is *robust* if it can make progress using answers from ℓ_2 -oracles with somewhat inaccurate inputs. The combination of stability and robustness is especially powerful. Together, these properties ensure that we can delay making small coordinate updates to inputs until they build up to a large cumulative update, and that we only get few large cumulative updates, enabling the use of coordinate-sparse update techniques. This approach of batching together small updates is known as *lazy* inverse updating. Obtaining further speed-ups using sketching also crucially relies on robustness. Because of robustness, we can afford to use sketching to estimate $\mathbf{x}^{(i)}$, as long as our estimates allow sufficiently accurate updates to the weights $\{r_e^{(i)}\}$.

The IPM of [CLS21] first achieved a running time of $\tilde{O}(n^{2+1/6} + n^\omega)$ by introducing a method with excellent stability and robustness, which in turn allowed them to implement a powerful inverse maintenance approach using lazy updates and sketching. Later, [Bra20] showed that the same running time can be obtained deterministically using only lazy updates, and finally [Jia+21] gave an improved running time of $\tilde{O}(n^{2+1/18} + n^\omega)$ using both lazy updates and sketching. The approach of [Jia+21] can be thought of as a two-level inverse maintenance, and the use of the randomized sketching techniques is crucial for them to efficiently implement the *query* operation of this data structure. It remains open if there exists any deterministic IPM that can run faster than $\tilde{O}(n^{2+1/6} + n^\omega)$.

Acceleration via width-reduction. In oracle-based optimization, there is a long history of developing *accelerated* methods, which reduce the iteration count compared to more basic approaches. This can be traced back to accelerated solvers for quadratic objectives [Lan52; HS52] and first-order acceleration for gradient Lipschitz functions ([Nes83] and earlier works by Nemirovski). Christiano et al. [Chr+11] developed an acceleration method for multiplicative weight methods that reduces the iteration count for solving ℓ_∞ regression with ℓ_2 -oracles from $\tilde{O}(\sqrt{n} \cdot \text{poly}(1/\epsilon))$ to $\tilde{O}(n^{1/3} \cdot \text{poly}(1/\epsilon))$. An alternative approach to acceleration for ℓ_∞ -regression can be obtained via the methods of Monteiro and Svaiter [MS13], and has also been a major research topic, but is beyond the scope of our discussion. For simplicity of our remaining discussion, we ignore ϵ dependencies. A rough outline of the MWU acceleration approach of Christiano et al. [Chr+11] is as follows: The MWU solves a sequence of ℓ_2 -oracle problems returning iterates $\mathbf{x}^{(i)}$. If we scale the problem so that $\|\mathbf{C}\mathbf{x}^* - \mathbf{d}\|_\infty \leq 1$, then weights ensure that (a) in each iteration, $\|\mathbf{C}\mathbf{x}^{(i)} - \mathbf{d}\|_\infty \lesssim \sqrt{n}$ and (b) after $T = \tilde{O}(\sqrt{n})$ iterations, $\tilde{\mathbf{x}} = \frac{1}{T} \sum_i \mathbf{x}^{(i)}$ has $\|\mathbf{C}\tilde{\mathbf{x}} - \mathbf{d}\|_\infty \leq 1 + \epsilon$. [Chr+11] made an important modification: if in some iteration we have $\|\mathbf{C}\mathbf{x}^{(i)} - \mathbf{d}\|_\infty \geq \rho \approx n^{1/3}$, then instead of using $\mathbf{x}^{(i)}$, we will adjust the weights $\{r_e^{(i)}\}$ in order to reduce the value of $\|\mathbf{C}\mathbf{x}^{(i')} - \mathbf{d}\|_\infty$ for future iterates $\mathbf{x}^{(i')}$. Using this method, an approximately optimal $\tilde{\mathbf{x}} = \frac{1}{T} \sum_i \mathbf{x}^{(i)}$ can be found in $T = \tilde{O}(n^{1/3})$ iterations. The parameter ρ measures the ℓ_∞ -norm $\|\mathbf{C}\mathbf{x}^{(i)} - \mathbf{d}\|_\infty$ of each iterate, sometimes known as the *width*, and the weight-adjustment steps of Christiano et al. are hence known as *width reduction steps*. When the oracle width can be reduced in this way, we will say our method is *width-reducible*. This acceleration has never been developed for ℓ_∞ -regression in the high-accuracy regime (i.e. running times that scale as $\text{polylog}(1/\epsilon)$), and whether this is possible is one of the major open questions in convex optimization.

Weight monotonicity in MWUs: an obstacle to sketching. Many MWU methods are designed to have an important property, which we call *weight monotonicity*. Concretely, in [Chr+11] and many other MWUs, the oracle weights $\{r_e^{(i)}\}$ are only growing. This often simplifies analyses greatly, and helps establish other properties including stability, robustness, and width-reducibility. Referring back to our oracle queries introduced above in (2), let us define $\tilde{\mathbf{x}}^{(i)} = \frac{1}{T} \sum_{j \leq i} \mathbf{x}^{(j)}$. Weight monotonicity arises because we choose the weights based on an overestimate of $|(\mathbf{C}\tilde{\mathbf{x}}^{(i)} - \mathbf{d})_e|$ given by $\gamma_i = \frac{1}{T} \sum_{j \leq i} |(\mathbf{C}\mathbf{x}^{(j)} - \mathbf{d})_e|$. In particular, choosing $r_e^{(i)} = \exp(\alpha \gamma_i)$ for some scaling factor α will ensure the weights only grow. As we will discuss later, *weight monotonicity* seems inherently incompatible with sketching, and thus we will need to develop a non-monotone MWU. Prior work by Madry [Mad13; Mad16] introduced non-monotone weights in a highly specialized IPM for unit-capacity maximum flow. This IPM of Madry has MWU-like properties and allows for some acceleration. The method has other drawbacks including low stability and robustness, but nonetheless inspired some of our design choices.

Prior inverse maintenance with acceleration. We are aware of a single prior work which combined lazy inverse updates with an accelerated MWU to obtain a running time of $\tilde{O}(n^{2+1/3} + n^\omega)$ for ℓ_p -regression [Adi+19]. This approach is relatively naive, falling short of the $\tilde{O}(n^{2+1/6} + n^\omega)$ running time which can be achieved using only lazy inverse updates.

1.3 Discussion of Techniques

The crucial algorithmic techniques we rely on for speeding up ℓ_∞ -regression are (a) acceleration, (b) lazy inverse updates, and (c) sketching. We can view each of these techniques as being en-

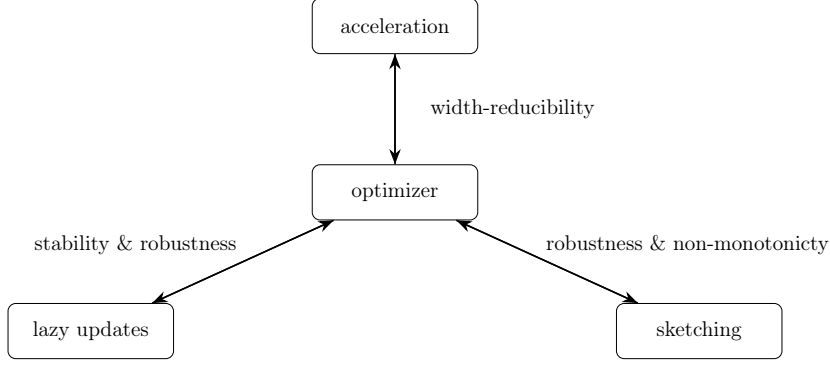


Figure 1: Algorithmic techniques and their requirements on our optimizer.

abled by different properties of the overall optimization approach. Our approach to acceleration is enabled by *width-reducibility*, while lazy updates require *stability* and *robustness*, and finally sketching requires *robustness* and *non-monotonicity*. This means we need to develop an MWU which simultaneously exhibits all these properties, i.e. it must be stable, robust, non-monotone, and width-reducible. In Figure 1, we summarize how our algorithmic techniques impose different requirements on our optimization approach. Again, for simplicity, in the remaining discussion of our results and prior work on this problem, we focus on the case $\omega = 2 + o(1)$.

We first discuss how to combine stability, robustness, and width-reducibility in a *monotone* MWU, which leads to a comparatively simple, deterministic algorithm using acceleration and lazy inverse updates, but no sketching.

Stability and robustness of a monotone, width-reducible MWU. [Adi+19] showed how to obtain stability, robustness, and width-reducibility together, with a monotone MWU. However, this work only established a weak notion of stability and hence comparatively slow running time of $\tilde{O}(n^{2+1/3})$. In contrast, one can show that by directly using stability and robustness in a monotone accelerated MWU, one can adapt the data structure approach of [Bra20] to achieve a running time of $\tilde{O}(n^{2+1/9})$, yielding a faster MWU.

Our first result Theorem E.1 is based on the observation that monotone MWU also enables a new, stronger notion of stability, which we call ℓ_3 -stability. This allows us to further reduce the number of lazy updates we make and lets us achieve a deterministic running time of $\tilde{O}(n^{2+1/12})$.

Non-monotone MWU - a key ingredient for sketching. As we described above, it is relatively easy to improve the running time of low-accuracy ℓ_∞ -regression among deterministic algorithms, by designing a monotone, robust, width-reducible MWU with a novel ℓ_3 -stability.

To further accelerate the algorithm by using a two-level inverse maintenance data structure, we need to use randomized sketching techniques to efficiently implement the query operation, which is required in every iteration of the MWU algorithm. Unfortunately, weight monotonicity is in conflict with sketching, because monotonicity arises from ignoring cancellations in $(\mathbf{C}\tilde{\mathbf{x}}^{(i)} - \mathbf{d})_e$ between different iterations.² In contrast, when using sketching, we want to crucially rely on cancellation between different iterations, as we sometimes overestimate $(\mathbf{C}\mathbf{x}^{(i)} - \mathbf{d})_e$ and sometimes underestimate it, but get it right on average. Because of this, we design an MWU with non-monotone weights. This in turn makes width-reducibility, robustness, and stability much harder to obtain.

²Recall that the final output of our MWU is the last averaged iterate $\tilde{\mathbf{x}}^{(T)}$.

To allow us to work with non-monotone weights and still obtain acceleration, we introduce a more delicate width-reduction scheme, inspired by [Mad16]. We also provide a tighter analysis of the sketching technique (it was named coordinate-wise embedding by [LSZ19; Jia+21]) that upper bounds its total noise across different iterations using martingale concentration inequalities. This tighter analysis is necessary to control the overall error introduced by the sketching technique in our MWU algorithm. We believe this tighter analysis could also provide a simpler analysis for the IPM results of [CLS21; Jia+21].

This new width-reduction approach in turn also requires us to estimate an ℓ_3 -norm associated with each iterate $\mathbf{x}^{(i)}$, and to do this quickly, we need to employ new sketching tools. To implement this approach, we also need an additional heavy-hitter sketch that allows us to identify which weights to adjust during width reduction.

Stability and robustness of a non-monotone, width-reducible MWU. Stability and robustness are crucial when we want to use lazy updates and sketching for inverse maintenance. Standard techniques for acceleration by width-reduction are unstable in the context of *non-monotone* MWU. Thus, to combine stability, width-reduction, and non-monotonicity, we have to further change our width-reduction strategy.

A central challenge is that width-reducibility is inherently in tension with the other properties. To simultaneously achieve stability and width-reducibility, we introduce a new and rather different approach to width-reduction, which we call *stable width-reduction*. This approach is more conservative than existing methods, and uses smaller width-reduction steps to achieve stability.

Combining width-reducibility with robustness is also difficult. Width-reduction relies on identifying too-large entries of the oracle outputs and making adjustments to the corresponding weights. But, robustness requires us to operate with inaccurate weights. We want to allow for weights that are inaccurate up to a factor $(1 \pm 1/\text{polylog}(n))$, and this is enough to completely change which oracle outputs are too large. In fact, we do not achieve general robust, but instead show that our method is robustness to (1) the errors induced by our specific lazy update scheme and (2) the errors induced by sketching.

Future perspectives. It remains open to design any algorithm for low-accuracy ℓ_∞ regression beyond $\tilde{O}(n^{2+1/22.5})$ when $\omega = 2 + o(1)$. We remark that if it were possible to use ℓ_3 -stability with the two-level data structure and an algorithm that converges in $n^{1/3}$ iterations, then we would achieve a runtime of $\tilde{O}(n^{2+1/48})$. However, the current techniques for inverse maintenance and acceleration are not sufficient to achieve $\tilde{O}(n^{2+o(1)} + n^\omega)$, which we believe would require substantially new techniques. On the other hand, even obtaining slight improvements in the runtime would require more robust acceleration and inverse maintenance frameworks which would be of independent interest.

In this paper, we analyze our algorithms in the RealRAM model. Establishing a similar analysis in finite precision arithmetic is an interesting open problem. Inverse maintenance-based IPM with finite precision arithmetic was studied by [GPV23].

We have demonstrated that acceleration techniques for MWU can be efficiently combined with inverse maintenance methods. For linear programming, no similar acceleration techniques exist and it is a major open problem to design these or rule out the possibility in various computational models. If acceleration can be achieved for linear programming, deploying it in conjunction with inverse maintenance will likely require techniques similar to those we introduce in this work.

2 Technical Overview

2.1 Deterministic MWU Algorithm via One-Level Inverse Maintenance

MWU methods reduce ℓ_∞ -regression problems to a sequence of ℓ_2 -minimization problems, which can be solved by solving systems of linear equations – or equivalently, applying the inverse of some matrix. More concretely, an MWU for finding approximate solutions to $\min_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{C}\mathbf{x} - \mathbf{d}\|_\infty$ requires us to repeatedly solve problems of the form

$$\min_{\Delta \in \mathbb{R}^d} \sum_e \mathbf{r}_e^{(i)} (\mathbf{C}\Delta - \mathbf{d})_e^2$$

across iterations $i = 1, \dots, T$. The exact solution to these minimization problems is given by

$$\Delta^{(i)} = (\mathbf{C}^\top \mathbf{R}^{(i)} \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{R}^{(i)} \mathbf{d}.$$

The multiplicative weight update method iteratively updates the weights using $\Delta^{(i)}$ and “penalizes” the coordinates e that have large $|\mathbf{C}\Delta^{(i)} - \mathbf{d}|_e$ by increasing their weights $\mathbf{r}_e^{(i+1)}$ in the next iteration. In the end the method outputs $\mathbf{x} = \sum_{i=1}^T \Delta^{(i)} / T$ as the approximate ℓ_∞ minimizer.

The cost of each iteration is dominated by the time required to solve the corresponding system of linear equations for $\Delta^{(i)}$ – or equivalently, applying the inverse of some matrix. If solving this sequence of systems of linear equations can be done faster than naively solving each system separately, then we can speed up the cost per iteration of the MWU algorithm, and hence make the algorithm faster. A similar problem of solving a sequence of systems of linear equations was studied for the IPM algorithms [CLS21; Bra20; Jia+21], and they achieved speed-ups by using lazy updates with *inverse maintenance* data structures. They could use lazy updates because the IPM algorithm satisfies a stability guarantee and a robustness guarantee. More precisely, (1) IPMs satisfy an ℓ_2 -stability guarantee that the ℓ_2 -norm of the relative changes between two iterations is bounded, i.e., $\|\frac{\mathbf{r}^{(i+1)} - \mathbf{r}^{(i)}}{\mathbf{r}^{(i)}}\|_2^2 \leq O(1)$. (2) IPMs are still correct if the system of linear equations is solved with coordinate-wise approximate weights $\bar{\mathbf{r}} \approx_\delta \mathbf{r}$ for some $\delta > 0$.

As it turns out, the *monotone* MWU algorithm is also inherently stable and robust, even with acceleration. We can therefore use coordinate-wise approximate weights $\bar{\mathbf{r}}^{(i)} \approx_\delta \mathbf{r}^{(i)}$ in each iteration, and only update $\bar{\mathbf{r}}_e^{(i)}$ when it differs from $\mathbf{r}_e^{(i)}$ by more than δ . This ensures that the approximate weights $\bar{\mathbf{r}}^{(i)}$ undergoes low-rank updates. We present a robust version of the known *accelerated* multiplicative weights update method for ℓ_∞ -regression from [Chr+11; Chi+13] below, where when solving the system of linear equations for $\Delta^{(i)}$ we use the approximate weights $\bar{\mathbf{r}}^{(i)}$.

Theorem 2.1 ([Chi+13]). *Let $0 < \epsilon < 1/2$ and $0 \leq \delta \leq \epsilon/6$. Algorithm 1 returns $\hat{\mathbf{x}}$ such that $\|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty \leq 1 + O(\epsilon)$ in $\tilde{O}(n^{1/3}\epsilon^{-7/3})$ iterations. Each iteration solves a linear system as specified in Line 9 of the algorithm.*

In fact, we can prove that this algorithm satisfies an even stronger stability guarantee – a quantitatively strong type of ℓ_3 -stability, namely

$$\sum_{i=1}^T \left\| \frac{\mathbf{r}^{(i+1)} - \mathbf{r}^{(i)}}{\mathbf{r}^{(i)}} \right\|_3^3 \leq O(n^{1/3}).$$

The ℓ_3 -stability guarantee allows for the following lazy-update scheme: for every ℓ , in every 2^ℓ iterations perform an update of size $O(2^{3\ell})$ to $\bar{\mathbf{r}}^{(i)}$.

Together with the one-level inverse maintenance of Brand, Nanongkai, and Saranurak [BNS19], this improves upon the previous best deterministic algorithm for low-accuracy ℓ_∞ regression that runs in $O(n^\omega + n^{2+1/6})$. We present a simplified version of the data structure below, and the formal version tailored to our application is in Section C.1.

Algorithm 1 Monotone Width Reduced MWU Algorithm

```

1: procedure MWU-SOLVER( $\epsilon, \mathbf{C}, \mathbf{d}$ )
2:    $\mathbf{w}^{(0,0)} \leftarrow \mathbf{1}_n, \quad \mathbf{x}^{(0)} \leftarrow \mathbf{0}_d$ 
3:    $\tau \leftarrow \Theta\left(\frac{n^{\frac{1}{3}}}{\epsilon^{\frac{1}{3}}} \log \frac{n}{\Psi_0}\right), \alpha \leftarrow \Theta\left(n^{-\frac{1}{2}+\eta} \epsilon^{\frac{1}{3}} \left(\log \frac{n}{\Psi_0}\right)^{-1}\right), \eta \leftarrow \frac{1}{6}$ 
4:    $T \leftarrow \alpha^{-1} \epsilon^{-2} \log n$ 
5:    $i \leftarrow 0, k \leftarrow 0$ 
6:   while  $i < T$  do
7:      $\mathbf{r}_e^{(i,k)} \leftarrow \mathbf{w}_e^{(i,k)} + \frac{\epsilon}{n} \|\mathbf{w}^{(i,k)}\|_1$ 
8:      $\bar{\mathbf{r}}^{(i,k)} \leftarrow \text{SELECTVECTOR}(\mathbf{r}^{(i,k)}, i+k, \delta) \quad \triangleright \bar{\mathbf{r}} \approx_{\delta} \mathbf{r}$ 
9:      $\Delta^{(i,k)} \leftarrow \arg \min_{\Delta \in \mathbb{R}^d} \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\mathbf{C}\Delta - \mathbf{d})_e^2 \quad \triangleright \Delta = (\mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{d}$ 
10:    if  $\|\mathbf{C}\Delta^{(i,k)} - \mathbf{d}\|_\infty \leq \tau$  then  $\triangleright$  primal step
11:       $\mathbf{w}^{(i+1,k)} \leftarrow \mathbf{w}^{(i,k)} (1 + \epsilon \alpha \|\mathbf{C}\Delta^{(i,k)} - \mathbf{d}\|_\infty)$ 
12:       $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \Delta^{(i,k)}$ 
13:       $i \leftarrow i + 1$ 
14:    else
15:      For all coordinates  $e$  with  $|\mathbf{C}\Delta^{(i,k)} - \mathbf{d}|_e \geq \tau$   $\triangleright$  width reduction step
16:         $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon) \mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{n} \|\mathbf{w}^{(i,k)}\|_1$ 
17:         $k \leftarrow k + 1$ 
18:  return  $\hat{\mathbf{x}} = \frac{\mathbf{x}^{(T)}}{T}$ 

```

Lemma 2.2 (One-level inverse maintenance, (Informal) Theorem 4.1 of [BNS19]). *There is a data structure that supports the following two operations to maintain the inverse of an $n \times n$ matrix M :*

- **Reset:** Reset M^{-1} to $(M + \Delta)^{-1}$, where Δ has k_0 non-zero entries. This operation can be done in $O(\mathcal{T}_{\text{mat}}(n, n, k_0))$ time.³
- **Query:** Output the vector $(M + \Delta)^{-1} \cdot v$ using the maintained M^{-1} and $M^{-1}v$, where Δ has at most n^{a_0} non-zero entries. This operation can be done in $O(n^{\omega a_0} + n^{1+a_0})$ time.

Runtime when $\omega = 2$. For simplicity, we only show the runtime of our algorithm when $\omega = 2$ in this section and omit polylogarithmic factors. Let us choose the parameter $a_0 = 3/4$, so that we perform a reset operation whenever we accumulate more than $n^{a_0} = n^{3/4}$ updates to $\bar{\mathbf{r}}$. From our low-rank update scheme under the ℓ_3 stability guarantee, this only happens in every $n^{1/4}$ iterations. So we perform a reset operation with cost $O(n^2)$ (since $\omega = 2$) in every $O(n^{1/4})$ iterations, and over the total $O(n^{1/3})$ iterations this gives a total reset time of $O(n^{2-1/4} \cdot n^{1/3}) = O(n^{2+1/12})$.

We perform a query operation in every iteration with cost $O(n^{2a_0} + n^{1+a_0}) = O(n^{1+3/4})$. Over all $O(n^{1/3})$ iterations this gives a total query time of $O(n^{1+3/4} \cdot n^{1/3}) = O(n^{2+1/12})$. Therefore, the total runtime is the sum of the reset time and the query time, which is $O(n^{2+1/12})$ as claimed in Theorem 1.1.

2.2 Randomized MWU Algorithm via Two-Level Inverse Maintenance

To further improve the runtime of the algorithm, we will use the following, more efficient two-level inverse maintenance data structure.

³ $\mathcal{T}_{\text{mat}}(n, r, m)$ denotes the time complexity of multiplying an $n \times r$ matrix with an $r \times m$ matrix.

Lemma 2.3 (Two-level inverse maintenance, (Informal) Theorem 4.2 of [BNS19]). *There is a data structure that supports the following three operations to explicitly maintain the inverse of an $n \times n$ matrix M . The algorithm achieves the goal via explicitly maintaining the inverse of an $n \times n$ matrix M_0 and implicitly maintaining the inverse of another $n \times n$ matrix M_1 that differs from M_0 on at most n^{a_0} entries, and the true matrix M always differ from M_1 on at most n^{a_1} entries where $a_1 \leq a_0$:*

- **Reset:** *Reset M_0^{-1} to $(M_0 + \Delta_0)^{-1}$, where Δ_0 has k_0 non-zero entries. This operation can be done in $\mathcal{T}_{\text{mat}}(n, n, k_0)$ time.*
- **Partial reset:** *Implicitly reset M_1^{-1} to $(M_1 + \Delta_1)^{-1}$, where Δ_1 has k_1 non-zero entries. This operation can be done in $\mathcal{T}_{\text{mat}}(n, n^{a_0}, k_1)$ time.*
- **Query:** *Output ℓ entries of the vector $M^{-1} \cdot v$ using the maintained M_0^{-1} , M_1^{-1} (implicitly). This operation can be done in $\mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, \max\{n^{a_1}, \ell\})$ time.*

The total runtime of the above data structure is the sum of its reset, partial reset, and query times. Let us now compare the query times of this two-level data structure with the one-level version. Observe that, the query time of the one-level data structure is n^{1+a_0} and that of the two-level data structure is better than n^{1+a_0} only if $\ell = o(n)$. In other words, we get an improvement via the two-level data structure only if we have an algorithm that does not require querying the entire maintained vector $M^{-1}v$.

So far, such an improvement via the two-level data structure has only been utilized, although in a complicated way, in the work of Jiang et al. [Jia+21] where they give a fast algorithm for linear programming by using the data structure within the robust interior point method framework and querying a *sketch* of the vector at every iteration. It is still an open problem if one can achieve their runtime of $\approx n^{2+1/18}$ via a deterministic algorithm and it is conjectured that improving the runtime either requires an improved data structure or, a more sophisticated “dimension reduction technique” to work with the algorithm.

Sketching and non-monotone MWU. Similar to [Jia+21], in our work we also query a sketch of the maintained vector in every iteration. More precisely, in each iteration we use a random matrix $\mathbf{S} \in \mathbb{R}^{n^{1/2+\eta} \times n}$ where η is the acceleration that we get, i.e., the total number of iterations is $O(n^{1/2-\eta})$, and we compute an approximate step $\mathbf{S}^\top \cdot \mathbf{S} \cdot (\mathbf{C}^\top \Delta^{(i,k)} - \mathbf{d})$. Using the coordinate-wise embedding guarantee of the random matrix \mathbf{S} , we can ensure that for each coordinate we have

$$\left(\mathbf{S}^\top \mathbf{S} (\mathbf{C}^\top \Delta^{(i,k)} - \mathbf{d}) \right)_e \approx (\mathbf{C}^\top \Delta^{(i,k)} - \mathbf{d})_e.$$

We now require to change Line 11 of Algorithm 1 to update the weights by

$$\mathbf{w}^{(i+1,k)} \leftarrow \mathbf{w}^{(i,k)} \left(1 + \epsilon \alpha \cdot \mathbf{S}^\top \mathbf{S} (\mathbf{C} \Delta^{(i,k)} - \mathbf{d}) \right).$$

Note that we lose monotonicity of the weights with this new primal step. We have to use this non-monotone update because the absolute values $|\mathbf{S}^\top \mathbf{S} (\mathbf{C}^\top \Delta^{(i,k)} - \mathbf{d})|$ would result in an error that is around the standard deviation of the estimator in every update of $\mathbf{w}^{(i,k)}$ ’s, and this would add up over iterates. Since the entire analysis of the MWU methods depends on tracking potentials which are functions of the weights, we would incur a large error. To circumvent this issue we require a version of the MWU method where the weights are not updated monotonically, and the random noise introduced by the sketching matrix \mathbf{S} can cancel out with each other across different coordinates e and across different iterations i .

Monotonicity is crucial in accelerating MWU methods and it is non-trivial to achieve accelerated rates without it. A few works in graph algorithms have been successful in obtaining

accelerated rates without monotonicity [Mad16; LS20] for specific algorithms. In this paper, we extend the algorithm of Madry [Mad16] to regression and obtain an algorithm with non-monotone updates that also converges in $n^{1/3}$ iterations and is robust (Refer to Appendix J for the complete algorithm and analysis).

Interior point methods directly control the solution quality of the last iterate. In contrast, MWU algorithms only measure the quality of the average of the primal iterates $\Delta^{(i,k)}$. As a result, our bound on the final solution requires a new MWU analysis that can handle cancellations between iterates of the errors arising from using sketching. We achieve this by developing a tighter analysis that upper bounds the sum of the sketching error over multiple iterations:

$$\sum_{i=0}^t \left(\left(\mathbf{S}^\top \mathbf{S} (\mathbf{C} \Delta^{(i)} - \mathbf{d}) \right)_e - (\mathbf{C} \Delta^{(i)} - \mathbf{d})_e \right) \lesssim \frac{\sqrt{nt}}{\sqrt{b}}.$$

We prove this bound using Freedman’s concentration bound for martingales. We also believe this tighter analysis can simplify the sketching analysis for the previous IPM papers [CLS21; LSZ19; Jia+21].

Stability and robustness of non-monotone MWU. The non-monotone MWU with standard width reduction steps is neither stable nor satisfies a low-rank update per iteration. We propose a new width reduction step that satisfies a low-rank update scheme which is sufficient for our data structure. Our steps, however, do not satisfy ℓ_2 stability, which is a sufficient condition for the low-rank update scheme. Instead of increasing all weights by a factor of $(1 + \epsilon)$ as in Line 16 of Algorithm 1, our new width reduction step increases a carefully selected set of weights. As a result, we can ensure that whenever we increase a large set of weights, we also increase the potential by a lot, so this event doesn’t happen very often. This helps ensure that weight updates from width-reduction steps occur on a similar “schedule” to weight updates from our primal update steps, and it allows us to efficiently handle both in the inverse maintenance data structure (Refer to Algorithm 3 and Appendix G for the complete algorithm and analysis). To efficiently find the coordinates e to perform width reduction on, we use an additional *heavy-hitter* data structure to identify these Δ_e exactly. We can only afford to find $n^{1/2+\eta}$ such coordinates in each iteration. This restriction on the number of coordinates restricts us to set η to be $1/10$, and our final iteration complexity is $n^{1/2-\eta} = n^{2/5}$ instead of $n^{1/3}$. The non-monotone algorithm also requires estimating a weighted ℓ_3 -norm of $\hat{\Delta}^{(i,k)}$ ’s for which we use an additional sketch from [WZ13].

Unlike the width reduction steps, the primal steps are stable, and they satisfy the ℓ_2 stability,

$$\left\| \frac{\mathbf{r}^{(i+1)} - \mathbf{r}^{(i)}}{\mathbf{r}^{(i)}} \right\|_2^2 \leq O(n^{2\eta}).$$

Given the ℓ_2 stability guarantee, we again use coordinate-wise approximate weights $\bar{\mathbf{r}}^{(i)} \approx_\delta \mathbf{r}^{(i)}$ in each primal step, and only update $\bar{\mathbf{r}}_e^{(i)}$ to be $\mathbf{r}_e^{(i)}$ if it differs from $\mathbf{r}_e^{(i)}$ by more than δ . This again guarantees a low-rank update scheme for the primal steps: for every ℓ , in every 2^ℓ iterations we only perform an update of size $O(2^{2\ell} \cdot n^{2\eta})$ to $\bar{\mathbf{r}}^{(i)}$.

It is non-trivial to show that the accelerated non-monotone MWU is robust under such coordinate-wise approximations to the weights. This is because we do not update the weights in every primal step, and we lazily update them in future iterations. We use an amortization argument to show that we can still gain enough changes in the required potentials even when we defer some updates to the future. However, this means our accelerated non-monotone MWU is only robust under the specific approximate weights $\bar{\mathbf{r}}_e^{(i)}$ that are updated to be $\mathbf{r}_e^{(i)}$ whenever

it differs too much from $\mathbf{r}_e^{(i)}$. We cannot guarantee robustness if in every iteration we choose an arbitrary coordinate-wise approximation unless we consider the unaccelerated algorithm, which was guaranteed in the IPM algorithms.

Runtime when $\omega = 2$. Finally, we sketch the time complexity of our non-monotone MWU algorithm using sketching when $\omega = 2$. For simplicity, we omit polylogarithmic factors. Using the two-level inverse maintenance data structure of Lemma 2.3, we perform a reset operation whenever we accumulate more than n^{a_0} updates to $\bar{\mathbf{r}}$, and by our low-rank update scheme under the ℓ_2 stability guarantee, this only happens in every $n^{a_0/2-\eta}$ iterations. Similarly, we perform a partial reset operation whenever we accumulate more than n^{a_1} updates to $\bar{\mathbf{r}}$, and this only happens in every $n^{a_1/2-\eta}$ iterations. Finally, note that our query time is bounded by $n^{a_0+a_1}$ since we always ensure that we query for at most $\ell = O(n^{1/2+\eta})$ coordinates in each iteration. So our total runtime over $T = n^{1/2-\eta}$ iterations is

$$\underbrace{T \cdot \frac{n^2}{n^{a_0/2-\eta}}}_{\text{reset}} + \underbrace{T \cdot \frac{n^{1+a_0}}{n^{a_1/2-\eta}}}_{\text{partial reset}} + \underbrace{T \cdot n^{a_0+a_1}}_{\text{reset}} = n^{2.5-a_0/2} + n^{1.5+a_0-a_1/2} + n^{0.5-\eta+a_0+a_1}.$$

Choosing the parameters $a_0 = 1 - \frac{1-2\eta}{9}$ and $a_1 = 1 - \frac{1-2\eta}{3}$, we have that the total runtime is bounded by $O(n^{2+1/18-\eta/9})$. Since we achieve an acceleration of $\eta = 1/10$ and $n^{1/2-\eta} = n^{2/5}$ iterations, this gives the claimed $O(n^{2+1/18-\eta/9}) = O(n^{2+1/22.5})$ time complexity of Theorem 1.2.

3 Fast Width-Reduced MWU Algorithms

In this section, we present the formal guarantees of our multiplicative weight update routines: a deterministic MWU algorithm with monotone weights (Algorithm 1) that is used in Theorem 1.1, and a randomized MWU algorithm with non-monotone weights and stable and robust steps (Algorithm 3) that is used in Theorem 1.2.

3.1 Lazy Update Procedure

We first present the SELECTVECTOR algorithm (Algorithm 2) from [LV21] that computes a coordinate-wise approximate vector $\bar{\mathbf{r}}$ of \mathbf{r} such that $\bar{\mathbf{r}}$ undergoes small updates.

We remark that the only difference between our algorithm and that of [LV21] is in Line 11 where we only include a coordinate e in S if \mathbf{w}_e is not being updated by a width reduction step between primal iterations $i - 2^\ell$ and i . This is due to a minor technicality of dealing with the two kinds of steps, primal and width reduction, in Algorithm 11 and 3. In all our algorithms, if we toggle a coordinate e in a width reduction step, then we always update the “lazy” approximate vector $\bar{\mathbf{r}}_e$ to be the same as \mathbf{r}_e , so the guarantees of the SELECTVECTOR algorithm still hold under this change in Line 11.

Algorithm 2 Compute a coordinate-wise approximate vector that undergoes small updates [LV21]

```

1: procedure SELECTVECTOR( $\mathbf{r}^{(i)}, i, \delta$ )
2:    $\triangleright$  This procedure stores all previous  $\mathbf{r}^{(0)}, \dots, \mathbf{r}^{(i-1)}$ , and the  $\bar{\mathbf{r}}$  in the previous iteration
3:   if  $i = 0$  then
4:     return  $\bar{\mathbf{r}} \leftarrow \mathbf{r}^{(0)}$ 
5:    $S \leftarrow \emptyset$ 
6:   for  $\ell = 0, 1, \dots, \log n$  do
7:     if  $i \equiv 0 \pmod{2^\ell}$  then
8:       if  $\ell = \log n$  then
9:          $S \leftarrow [n]$ 
10:      else
11:         $S \leftarrow S \cup \{e : |\ln(\frac{\mathbf{r}_e^{(i)}}{\mathbf{r}_e^{(i-2^\ell)}})| \geq \frac{\delta}{2 \log n} \text{ and } \text{LASTWIDTH}(i, e) \leq i - 2^\ell\}$ 
12:         $\triangleright \text{LASTWIDTH}(i, e) \leq i$  is the last primal step during which a width
        reduction step updates  $w_e$ 
13:       $\bar{\mathbf{r}}_e \leftarrow \mathbf{r}_e^{(i)}$  for all  $e \in S$ 
14:      return  $\bar{\mathbf{r}}$ 

```

3.2 Monotone Multiplicative Weights Update Algorithm

We have already presented the convergence guarantees of Algorithm 1 in Theorem 2.1. We now add the stability guarantees that we use to prove the guarantees of our fast deterministic algorithm. The analysis of the algorithm is in Appendix F and the stability guarantees are in Appendix H.1.

Lemma 3.1 (Stability bound of ℓ_3 norm over all primal iterations). *Let k_i denote the number of width reduction steps taken by the algorithm when the i^{th} primal step is being executed. Then over all T primal steps of Algorithm 1, we have*

$$\sum_{i=0}^{T-1} \sum_{e \in S_i} \left(\frac{\mathbf{r}_e^{(i+1, k_i)} - \mathbf{r}_e^{(i, k_i)}}{\mathbf{r}_e^{(i, k_i)}} \right)^3 \leq \tilde{O}(\alpha^2 n) = \tilde{O}(n^{1/3} \epsilon^{2/3}).$$

Here S_i is the set of coordinates e at primal iteration i such that $\mathbf{r}_e^{(i+1, k_i)} \geq \mathbf{r}_e^{(i, k_i)} (1 + 3\epsilon\alpha)^4$.

Lemma 3.2 (Stability bound of ℓ_3 norm over all width reduction iterations). *Let i_k denote the number of primal steps taken before the execution of the k^{th} width reduction step. Then, over all K width reduction steps of Algorithm 1, we have*

$$\sum_{k=0}^{K-1} \left(\frac{\mathbf{r}_e^{(i_k, k+1)} - \mathbf{r}_e^{(i_k, k)}}{\mathbf{r}_e^{(i_k, k)}} \right)^3 \leq \tilde{O}(n^{1/3}).$$

3.3 Algorithm with Non-Monotone Weights, Stability and Robustness

We now give our main algorithm which can be used with our two-level inverse maintenance data structure. Algorithm 3 updates the weights in a non-monotone way, and additionally has stable primal and width reduction steps. It is also compatible with sketching as required by the data structure. We can prove the following guarantees.

⁴We note that it is sufficient to consider these sets S_i 's since any change that is smaller than the ones captured here can happen only $\tilde{O}(1)$ times.

Theorem 3.3. For $\eta \leq 1/10$, with probability $1-1/n^3$, Algorithm 3 with input $(\begin{bmatrix} \mathbf{C} \\ -\mathbf{C} \end{bmatrix}, \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix}, \epsilon)$ finds $\hat{\mathbf{x}} \in \mathbb{R}^n$ such that $\|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty \leq 1 + O(\epsilon)$ in at most $\tilde{O}(n^{1/2-\eta}\epsilon^{-4})$ iterations. Furthermore, the algorithm satisfies the following extra guarantees:

1. In the width reduction step of the algorithm, the algorithm only requires to find at most $\tilde{O}(n^{1/2+\eta})$ large coordinates per iteration.
2. The algorithm satisfies the following low-rank update scheme: There are at most $\frac{T+K}{2^\ell}$ number of iterations where $\bar{\mathbf{r}}$ receives an update of rank $\tilde{O}_\epsilon(n^{1/5}2^\ell)$.

In order to get Algorithm 3 we begin by extending the graph based algorithms of [Mad16] to ℓ_∞ -regression. A direct extension (Algorithm 10, analysis included in Appendix J) does not have stable width reduction steps. We therefore design a new set of width reduction steps (Algorithm 11, Appendix G.2), which necessitates a new analysis for bounding the number of such steps. We then additionally add sketching to the primal steps to get the final algorithm which is analyzed in Appendix G.3.

Organization of Appendix

The appendix contains detailed proofs of our algorithms. In Appendix A we give some preliminaries and basic results we use for our proofs. In Appendix B we prove the guarantees of our low-rank update scheme under ℓ_2 and ℓ_3 stability, given by the subroutine SELECTVECTOR in Algorithm 3. Further in Appendix C we provide the guarantees for all the data structures required to implement our final algorithms. In Appendix D we show how to implement our randomized algorithm (Algorithm 3) using the data structures from Appendix C, and prove Theorem 1.2. In Appendix E we show how to implement our deterministic MWU algorithm (Algorithm 1) using the data structures from Appendix C, and prove Theorem 1.1. In Appendix F and G we prove the guarantees of Algorithm 1 and Algorithm 3 respectively. In Appendix H we prove the stability guarantees of Algorithms 1 and 3. In Appendix I we provide the missing proofs for standard results from Appendix A and C. Finally, in Appendix J (optional) we provide the analysis of a non-monotone MWU that is robust but not stable and it has $O_\epsilon(n^{1/3})$ iterations. This analysis is included for completeness, and is not required to prove our results.

Algorithm 3 Accelerated MWU algorithm with non-monotone weights and stable and robust steps

```

1: procedure MWU-NONMONOTONEROBUST( $\tilde{\mathbf{C}}, \tilde{\mathbf{d}}, \epsilon$ )
2:    $\mathbf{w}^{(0,0)} \leftarrow \mathbf{1}_{2n}, \quad \bar{\mathbf{r}}^{(0,0)} \leftarrow \mathbf{r}^{(0,0)} \leftarrow (1 + \epsilon)\mathbf{1}_{2n}, \quad \mathbf{x}^{(0)} \leftarrow \mathbf{0}_d$ 
3:    $\alpha \leftarrow \tilde{\Theta}(n^{-1/2+\eta\epsilon})$ 
4:    $\tau \leftarrow \tilde{\Theta}(n^{1/2+\eta\epsilon^{-4}}), \quad \rho \leftarrow \tilde{\Theta}(n^{1/2-3\eta\epsilon^{-2}})$ 
5:    $T \leftarrow \alpha^{-1}\epsilon^{-2} \ln n$ 
6:    $i, k = 0$ 
7:    $b \leftarrow \tilde{\Theta}(n^{1/2+\eta\epsilon^{-2}})$ 
8:   Let  $\mathbf{S}^{(0)}, \mathbf{S}^{(1)}, \dots, \mathbf{S}^{(T-1)} \in \mathbb{R}^{b \times 2n}$  be random matrices as described in Lemma G.8.
9:   while  $i < T$  do
10:      $\Delta^{(i,k)} \leftarrow (\tilde{\mathbf{C}}^\top \bar{\mathbf{R}}^{(i,k)} \tilde{\mathbf{C}})^{-1} \tilde{\mathbf{C}}^\top \bar{\mathbf{R}}^{(i,k)} \tilde{\mathbf{d}} \quad \triangleright \Delta^{(i,k)} = \arg \min_{\Delta} \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e^2$ 
11:      $\mathbf{u}^{(i,k)} \leftarrow \tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}$ 
12:      $\hat{\mathbf{u}}^{(i,k)} \leftarrow (\bar{\mathbf{R}}^{(i,k)})^{-1/2} \cdot (\mathbf{S}^{(i)})^\top \mathbf{S}^{(i)} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}$ 
13:      $\Psi(\bar{\mathbf{r}}^{(i,k)}) \leftarrow \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\mathbf{u}_e^{(i,k)})^2$ 
14:     if  $\sum_e \bar{\mathbf{r}}_e^{(i,k)} |\mathbf{u}_e^{(i,k)}|^3 \leq C_3 \rho \Psi(\bar{\mathbf{r}}^{(i,k)})$  then  $\triangleright$  primal step
15:        $\mathbf{w}^{(i+1,k)} \leftarrow \mathbf{w}^{(i,k)} \left( 1 + \epsilon \bar{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)} \right), \quad \bar{\alpha}_e^{(i,k)} = \begin{cases} \alpha \cdot (1 + \epsilon \alpha \hat{\mathbf{u}}_e^{(i,k)}) & \text{if } \hat{\mathbf{u}}_e^{(i,k)} \geq 0 \\ \alpha / (1 - \epsilon \alpha \hat{\mathbf{u}}_e^{(i,k)}) & \text{else} \end{cases}$ 
16:        $\mathbf{r}^{(i+1,k)} \leftarrow \mathbf{w}^{(i+1,k)} + \frac{\epsilon}{2n} \sum_e \mathbf{w}_e^{(i+1,k)}$ 
17:        $\bar{\mathbf{r}}^{(i+1,k)} \leftarrow \text{SELECTVECTOR}(\mathbf{r}^{(i+1,k)}, i+1, \delta)$   $\triangleright$  Algorithm 2
18:        $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \Delta^{(i,k)}$ 
19:        $i \leftarrow i + 1$ 
20:     else if  $\sum_e \bar{\mathbf{r}}_e^{(i,k)} |\mathbf{u}_e^{(i,k)}|^3 \geq C_3^{-1} \rho \Psi(\bar{\mathbf{r}}^{(i,k)})$  then  $\triangleright$  width reduction step
21:       Let  $S$  be the set of coordinates  $e$  such that  $|\mathbf{u}_e^{(i,k)}| \geq \rho/(2C_3)$ 
22:        $H \subseteq S$  be maximal subset such that  $\sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)})$ 
23:       if  $H \neq S$  then
24:         Pick any  $\bar{e} \in S \setminus H$ .
25:         For all  $e \in H \cup \{\bar{e}\}$ ,  $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon) \mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n} \Phi(\mathbf{w}^{(i,k)})$ 
26:          $\mathbf{r}^{(i,k+1)} \leftarrow \mathbf{w}^{(i,k+1)} + \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i,k+1)})$ 
27:         For all  $e \in H \cup \{\bar{e}\}$ ,  $\bar{\mathbf{r}}_e^{(i,k+1)} \leftarrow \mathbf{r}_e^{(i,k+1)}$ 
28:       else
29:         for  $\zeta = \rho, 2\rho, 4\rho, \dots, 2^{c_\rho} \rho$  do
30:            $\triangleright c_\rho$  is defined to be the smallest integer  $c$  that satisfies  $2^c \rho \geq \sqrt{n/\epsilon}$ 
31:           Define the set  $H_\zeta = \{e \in H \mid |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \in [\zeta, 2\zeta]\}$ .
32:           If  $\sum_{e \in H_\zeta} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq \frac{\rho \Psi(\bar{\mathbf{r}}^{(i,k)})}{\log(\frac{n}{\epsilon\rho})}$ , set  $\zeta^* \leftarrow \zeta$ , and break.
33:           For all  $e \in H_{\zeta^*}$ ,  $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon) \mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n} \Phi(\mathbf{w}^{(i,k)})$ 
34:            $\mathbf{r}^{(i,k+1)} \leftarrow \mathbf{w}^{(i,k+1)} + \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i,k+1)})$ 
35:           For all  $e \in H_{\zeta^*}$ ,  $\bar{\mathbf{r}}_e^{(i,k+1)} \leftarrow \mathbf{r}_e^{(i,k+1)}$ 
36:          $k \leftarrow k + 1$ 
37:   return  $\mathbf{x}^{(T)}/T$ 

```

References

- [ABS21] D. Adil, B. Bullins, and S. Sachdeva. “Unifying width-reduced methods for quasi-self-concordant optimization”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 19122–19133 (cit. on p. 4).
- [Adi+19] D. Adil, R. Kyng, R. Peng, and S. Sachdeva. “Iterative refinement for ℓ_p -norm regression”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2019, pp. 1405–1424 (cit. on pp. 4, 6, 7, 77).
- [Adi+24] D. Adil, R. Kyng, R. Peng, and S. Sachdeva. “Fast algorithms for ℓ_p -regression”. In: *Journal of the ACM* 71.5 (2024), pp. 1–45 (cit. on pp. 4, 39).
- [AHK12] S. Arora, E. Hazan, and S. Kale. “The Multiplicative Weights Update Method: A Meta-Algorithm and Applications”. In: *Theory of Computing* 8.6 (2012), pp. 121–164 (cit. on p. 4).
- [BCS97] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*. Vol. 315. Springer Science & Business Media, 1997 (cit. on p. 21).
- [BN51] G. W. Brown and J. V. Neumann. “6. SOLUTIONS OF GAMES BY DIFFERENTIAL EQUATIONS”. In: *Contributions to the Theory of Games (AM-24), Volume I*. Ed. by H. W. Kuhn and A. W. Tucker. Princeton University Press, 1951, pp. 73–80. ISBN: 978-1-4008-8172-7 (cit. on p. 4).
- [BNS19] J. v. d. Brand, D. Nanongkai, and T. Saranurak. “Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds”. In: *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2019, pp. 456–480 (cit. on pp. 4, 9, 10, 11, 25, 26).
- [Bra20] J. Brand. “A Deterministic Linear Program Solver in Current Matrix Multiplication Time”. In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2020, pp. 259–278 (cit. on pp. 4, 5, 7, 9).
- [Bra21] J. van den Brand. “Unifying Matrix Data Structures: Simplifying and Speeding up Iterative Algorithms”. In: *Symposium on Simplicity in Algorithms (SOSA)*. SIAM. 2021, pp. 1–13 (cit. on p. 26).
- [Bul18] B. Bullins. “Fast Minimization of Structured Convex Quartics”. In: *arXiv preprint arXiv:1812.10349* (2018). arXiv: 1812.10349 (cit. on p. 4).
- [Car+20] Y. Carmon, A. Jambulapati, Q. Jiang, Y. Jin, Y. T. Lee, A. Sidford, and K. Tian. “Acceleration with a ball optimization oracle”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19052–19063 (cit. on p. 4).
- [Chi+13] H. H. Chin, A. Madry, G. L. Miller, and R. Peng. “Runtime guarantees for regression problems”. In: *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*. 2013, pp. 269–282 (cit. on pp. 4, 9).
- [Chr+11] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. “Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs”. In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 273–282 (cit. on pp. 4, 6, 9).
- [CLS21] M. B. Cohen, Y. T. Lee, and Z. Song. “Solving linear programs in the current matrix multiplication time”. In: *Journal of the ACM (JACM)* 68.1 (2021), pp. 1–39 (cit. on pp. 4, 5, 8, 9, 12).

- [EV19] A. Ene and A. Vladu. “Improved Convergence for ℓ_{inf}^t and ℓ_1 Regression via Iteratively Reweighted Least Squares”. In: *Proceedings of Machine Learning Research* 97 (2019) (cit. on p. 4).
- [Fre75] D. A. Freedman. “On tail probabilities for martingales”. In: *the Annals of Probability* (1975), pp. 100–118 (cit. on p. 60).
- [GPV23] M. Ghadiri, R. Peng, and S. S. Vempala. “The bit complexity of efficient continuous optimization”. In: *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2023, pp. 2059–2070 (cit. on p. 8).
- [HS52] M. R. Hestenes and E. Stiefel. “On the Convergence of the Conjugate Gradient Method for Singular Linear Operator Equations”. In: *J. Research Nat. Bur. Standards* 49 (1952), pp. 409–436 (cit. on p. 6).
- [Jia+20] H. Jiang, T. Kathuria, Y. T. Lee, S. Padmanabhan, and Z. Song. “A faster interior point method for semidefinite programming”. In: *FOCS*. 2020 (cit. on p. 21).
- [Jia+21] S. Jiang, Z. Song, O. Weinstein, and H. Zhang. “A faster algorithm for solving general LPs”. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 823–832 (cit. on pp. 4, 5, 8, 9, 11, 12, 26).
- [JL84] W. B. Johnson and J. Lindenstrauss. “Extensions of Lipschitz mappings into a Hilbert space”. In: *Contemporary mathematics* 26.189-206 (1984), p. 1 (cit. on p. 27).
- [Kan+11] D. M. Kane, J. Nelson, E. Porat, and D. P. Woodruff. “Fast moment estimation in data streams in optimal space”. In: *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 2011, pp. 745–754 (cit. on p. 28).
- [Kar84] N. Karmarkar. “A New Polynomial-Time Algorithm for Linear Programming”. In: *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*. 1984, pp. 302–311 (cit. on p. 4).
- [Kel+14] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. “An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations”. In: *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2014, pp. 217–226 (cit. on p. 4).
- [Lan52] C. Lanczos. “Solution of Systems of Linear Equations by Minimized Iterations”. In: *J. Res. Nat. Bur. Standards* 49.1 (1952), pp. 33–53 (cit. on p. 6).
- [LS15] Y. T. Lee and A. Sidford. “Efficient inverse maintenance and faster algorithms for linear programming”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 230–249 (cit. on p. 4).
- [LS20] Y. P. Liu and A. Sidford. “Faster energy maximization for faster maximum flow”. In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 2020, pp. 803–814 (cit. on p. 12).
- [LSZ19] Y. T. Lee, Z. Song, and Q. Zhang. “Solving empirical risk minimization in the current matrix multiplication time”. In: *Conference on Learning Theory*. PMLR. 2019, pp. 2140–2157 (cit. on pp. 8, 12, 55).
- [LV21] Y. T. Lee and S. S. Vempala. “Tutorial on the robust interior point method”. In: *arXiv preprint arXiv:2108.04734* (2021) (cit. on pp. 4, 13, 14, 21).
- [Mad13] A. Madry. “Navigating Central Path with Electrical Flows: From Flows to Matchings, and Back”. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 2013, pp. 253–262 (cit. on p. 6).

- [Mad16] A. Madry. “Computing maximum flow with augmenting electrical flows”. In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2016, pp. 593–602 (cit. on pp. 6, 8, 12, 15, 42).
- [MS13] R. D. C. Monteiro and B. F. Svaiter. “An Accelerated Hybrid Proximal Extragradient Method for Convex Optimization and Its Implications to Second-Order Methods”. In: *SIAM Journal on Optimization* 23.2 (2013), pp. 1092–1125. ISSN: 1052-6234 (cit. on pp. 4, 6).
- [Nes83] Y. Nesterov. “A Method for Solving the Convex Programming Problem with Convergence Rate $o(1/K^2)$ ”. In: *Dokl Akad Nauk SSSR* 269 (1983), p. 543 (cit. on p. 6).
- [NN89] Y. E. Nesterov and A. Nemirovskii. “Self-Concordant Functions and Polynomial-Time Methods in Convex Programming”. In: *Report, Central Economic and Mathematical Institute, USSR Acad. Sci* (1989) (cit. on p. 4).
- [NN94] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994 (cit. on p. 4).
- [Pag13] R. Pagh. “Compressed matrix multiplication”. In: *ACM Transactions on Computation Theory (TOCT)* 5.3 (2013), pp. 1–17 (cit. on p. 28).
- [Ren88] J. Renegar. “A Polynomial-Time Algorithm, Based on Newton’s Method, for Linear Programming”. In: *Mathematical programming* 40.1 (1988), pp. 59–93 (cit. on p. 4).
- [She13] J. Sherman. “Nearly Maximum Flows in Nearly Linear Time”. In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 2013, pp. 263–269 (cit. on p. 4).
- [ST18] A. Sidford and K. Tian. “Coordinate methods for accelerating ℓ_∞ regression and faster approximate maximum flow”. In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 922–933 (cit. on p. 4).
- [Vai89] P. M. Vaidya. “Speeding-up Linear Programming Using Fast Matrix Multiplication”. In: *30th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1989, pp. 332–337 (cit. on p. 4).
- [WZ13] D. Woodruff and Q. Zhang. “Subspace embeddings and ℓ_p -regression using exponential random variables”. In: *Conference on Learning Theory*. PMLR. 2013, pp. 546–567 (cit. on pp. 12, 27).

A Preliminaries

Basic notations. For any vectors \mathbf{x} and \mathbf{y} with non negative entries, and $\delta > 0$ we use $\mathbf{x} \approx_\delta \mathbf{y}$ to imply that for all coordinates i , we have $e^{-\delta} \mathbf{y}_i \leq \mathbf{x}_i \leq e^\delta \mathbf{y}_i$. We use $\tilde{O}(\cdot)$ and $\tilde{\Theta}(\cdot)$ to hide poly log n factors, and we use $\tilde{O}_\epsilon(\cdot)$ and $\tilde{\Theta}_\epsilon(\cdot)$ to additionally hide poly(ϵ^{-1}) factors.

Given any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we use $\mathbf{x} \cdot \mathbf{y} \in \mathbb{R}^n$ to denote the coordinate-wise multiplication of the two vectors, i.e., its i -th entry is $\mathbf{x}_i \cdot \mathbf{y}_i$. Similarly, we also use other scalar operations on vectors to denote coordinate-wise operations.

For any vector $\mathbf{r} \in \mathbb{R}^n$, we use the capital letter $\mathbf{R} \in \mathbb{R}^{n \times n}$ to denote a diagonal matrix whose diagonal entries are \mathbf{r} .

Potential functions. In this paper we consider a fixed problem $\min_x \|\mathbf{C}\mathbf{x} - \mathbf{d}\|_\infty$ and assume that this has optimum objective value 1. We define the following two potential functions for weights \mathbf{w} and \mathbf{r} such that $\mathbf{r} = \mathbf{w} + \frac{\epsilon}{n} \|\mathbf{w}\|_1$:

$$\Phi(\mathbf{w}) \stackrel{\text{def}}{=} \|\mathbf{w}\|_1 \quad (3)$$

$$\Psi(\mathbf{r}) \stackrel{\text{def}}{=} \min_{\Delta \in \mathbb{R}^d} \sum_e \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2. \quad (4)$$

The two potentials satisfy the following three lemmas. Their proofs are standard, and we defer them to Section I.

The first lemma shows how the two potentials are related.

Lemma A.1. *Let $\mathbf{w} \geq 0, \mathbf{r}, \bar{\mathbf{r}}$, such that $\forall e, \mathbf{r}_e = \mathbf{w}_e + \frac{\epsilon}{n} \|\mathbf{w}\|_1$, and $\bar{\mathbf{r}} \approx_\delta \mathbf{r}$. Then, $\Psi(\bar{\mathbf{r}}) \approx_\delta \Psi(\mathbf{r})$, and $\Psi(\bar{\mathbf{r}}) \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w})$.*

In our algorithms we have the following lower bound on the initial Ψ potential.

Lemma A.2. *If $\mathbf{w}^{(0,0)} = 1$ and $\mathbf{r}^{(0,0)} = \mathbf{w}^{(0,0)} + \frac{\epsilon}{n} \cdot \Phi(\mathbf{w}^{(0,0)})$, then we have $\Psi(\mathbf{r}^{(0,0)}) \geq \Psi_0 \stackrel{\text{def}}{=} \min\{1, \mathbf{d}^\top (\mathbf{I} - \mathbf{C}^\top (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}) \mathbf{d}\}$.*

The last lemma provides a lower bound on the Ψ potential after a small perturbation to the weights \mathbf{r} .

Lemma A.3. *Let $\Psi(\mathbf{r}) = \min_\Delta \sum_e \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2$. For any $\mathbf{r}', \mathbf{r} \geq 0$ that satisfies $|\mathbf{r}'_e - \mathbf{r}_e| \leq \mathbf{r}'_e$ for all e , we have*

$$\Psi(\mathbf{r}') \geq \Psi(\mathbf{r}) + \sum_e \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}'_e} \right) \mathbf{r}_e (\mathbf{C}\tilde{\Delta} - \mathbf{d})_e^2,$$

where $\tilde{\Delta} := \arg \min_\Delta \sum_e \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2$.

Primal iterate and width iterate of MWU algorithms. We will use i to denote primal iterates and k to denote the width reduction iterations in our multiplicative weight update (mwu) algorithms. We use $\hat{\mathbf{u}}$ to denote the vector \mathbf{u} after applying sketching, and $\bar{\mathbf{u}}$ to denote an approximation to the vector \mathbf{u} . For the (mwu) algorithms, we would use i_k to denote the number of primal steps executed when the k^{th} width step is being taken, i.e., the k^{th} width step is from (i_k, k) to $(i_k, k+1)$, and we use k_i to denote the number of width reduction steps executed when the i^{th} primal step is taken, i.e., the i^{th} primal step is from (i, k_i) to $(i+1, k_i)$.

For any primal step i and any coordinate e , we define $\text{LASTWIDTH}(i, e)$ to be the largest $i' \leq i$ such that the algorithm executed a width reduction step from (i', k) to $(i', k+1)$ during which the weight of e is updated, i.e., $\mathbf{w}_e^{(i', k+1)} \neq \mathbf{w}_e^{(i', k)}$.

Fast matrix multiplication. We use $\mathcal{T}_{\text{mat}}(n, r, m)$ to denote the time complexity required to compute the product of an $n \times r$ matrix with an $r \times m$ matrix.

In our proofs we will frequently use the following fact. See e.g. [BCS97] for the basic properties of fast matrix multiplication exponents.

Fact A.4. $\mathcal{T}_{\text{mat}}(n, r, m) = O(\mathcal{T}_{\text{mat}}(n, m, r)) = O(\mathcal{T}_{\text{mat}}(m, n, r))$.

Definition A.5 (Fast matrix multiplication exponent). *For any β , define a function $\omega_\beta(x)$ to be the minimum value such that $\mathcal{T}_{\text{mat}}(n, n^x, n^\beta) = n^{\omega_\beta(x)+o(1)}$.*

With an abuse of notation we also define the function $\omega(x) = \omega_1(x)$, and define the value $\omega = \omega(1)$.

We also define $\alpha_ \in \mathbb{R}_+$ to be the dual exponent of matrix multiplication, i.e., $\omega(\alpha_*) = 2$.⁵*

We will also use the following fact about convexity. We present a proof (deferred to Section I) that generalizes the proof of Lemma 3.6 of [Jia+20].

Fact A.6 (Convexity). *For any β , $\omega_\beta(x)$ is convex in x .*

Fact A.7 (Upper bound of $\mathcal{T}_{\text{mat}}(n, n, r)$). *For any $r \leq n$,*

$$\mathcal{T}_{\text{mat}}(n, n, r) \leq n^{2+o(1)} + r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

B Low Rank Update Scheme under Stability Guarantees

B.1 Low Rank Update under ℓ_2 Stability

Algorithm 2 is the same as [LV21], and it satisfies the following lemma.

Lemma B.1 (Low-rank update scheme under ℓ_2 stability, Lemma 19 of [LV21]). *If we have the guarantee*

$$\sum_e \ln \left(\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} \right)^2 \leq \zeta,$$

then the Algorithm 2 outputs a vector $\bar{\mathbf{r}} \approx_\delta \mathbf{r}^{(i)}$ in each iteration, and the approximate vector $\bar{\mathbf{r}}$ undergoes a update of size $O((\frac{\log n}{\delta})^2 \cdot \zeta \cdot 2^\ell)$ in every 2^ℓ iterations for every $\ell \in [0 : \log T]$.

Next we show that the robust ℓ_2 stability guarantee also generates a δ -approximate sequence with low-rank updates.

Lemma B.2 (Low-rank update scheme under robust ℓ_2 stability). *If the sequence $\mathbf{r}_e^{(0)}, \dots, \mathbf{r}_e^{(T)}$ satisfies the following guarantee: There exists another sequence $\tilde{\mathbf{r}}_e^{(0)}, \dots, \tilde{\mathbf{r}}_e^{(T)}$ such that*

1.

$$\sum_e \ln \left(\frac{\tilde{\mathbf{r}}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} \right)^2 \leq \zeta, \quad \forall i \in [0 : T],$$

2. $\forall t \leq t' \in [T], \forall e$, with probability $1 - 1/n^4$,

$$\left| \sum_{i=t'-t}^{t'} \ln \left(\frac{\tilde{\mathbf{r}}_e^{(i)}}{\mathbf{r}_e^{(i)}} \right) \right| \leq \frac{\delta}{10 \log n}.$$

⁵It's common in the literature to use α to denote the dual exponent of matrix multiplication. We use α_* here because we will use α to denote the “step size” of accelerated MWU.

Then the Algorithm 2 outputs a vector $\bar{\mathbf{r}} \approx_{\delta} \mathbf{r}^{(i)}$ in each iteration, and the approximate vector $\bar{\mathbf{r}}$ undergoes an update of size $O((\frac{\log n}{\delta})^2 \cdot \zeta \cdot 2^{2\ell})$ in every 2^ℓ iterations for every $\ell \in [0 : \log T]$.

Proof. Consider a fixed iteration i . We first show that in the i -th iteration $e^{-\delta} \mathbf{r}_e^{(i)} \leq \bar{\mathbf{r}}_e \leq e^{\delta} \mathbf{r}_e^{(i)}$ for any $e \in [n]$. Let i' be the iteration when $\bar{\mathbf{r}}_e$ was last updated. We can write $i' = i_0 < i_1 < i_2 < \dots < i_s = i$ such that $i_{j+1} - i_j$ is a power of 2 and $i_{j+1} - i_j$ divides i_{j+1} , and $|s| \leq 2 \log n$. Hence, we have that

$$\frac{\mathbf{r}_e^{(i)}}{\bar{\mathbf{r}}_e} = \frac{\mathbf{r}_e^{(i_s)}}{\mathbf{r}_e^{(i_0)}} = \prod_{j=0}^{s-1} \frac{\mathbf{r}_e^{(i_{j+1})}}{\mathbf{r}_e^{(i_j)}} = \exp\left(\sum_{j=0}^{s-1} \ln\left(\frac{\mathbf{r}_e^{(i_{j+1})}}{\mathbf{r}_e^{(i_j)}}\right)\right) \leq \exp(\delta),$$

where in the fourth step we used that since $\bar{\mathbf{r}}_e$ is not updated since step i' , we have $|\ln(\frac{\mathbf{r}_e^{(i_{j+1})}}{\mathbf{r}_e^{(i_j)}})| \leq \frac{\delta}{2 \log n}$. Similarly we also have $\frac{\mathbf{r}_e^{(i)}}{\bar{\mathbf{r}}_e} \geq \exp(-\delta)$.

Next we bound the size of the update after every 2^ℓ iterations. Let i be any iteration where $i \equiv 0 \pmod{2^\ell}$. We denote the set that is being updated as $S_\ell := \{e : |\ln \frac{\mathbf{r}_e^{(i)}}{\mathbf{r}_e^{(i-2^\ell)}}| \geq \frac{\delta}{2 \log n}\}$. Wlog assume that $\ln \frac{\mathbf{r}_e^{(i)}}{\mathbf{r}_e^{(i-2^\ell)}} \geq 0$. Using the second property of the sequence that $\left|\sum_{j=i-2^\ell+1}^i \ln\left(\frac{\tilde{\mathbf{r}}_e^{(j)}}{\mathbf{r}_e^{(j)}}\right)\right| \leq \frac{\delta}{10 \log n}$, we have

$$\frac{\mathbf{r}_e^{(i)}}{\mathbf{r}_e^{(i-2^\ell)}} = \prod_{j=i-2^\ell}^{i-1} \frac{\mathbf{r}_e^{(j+1)}}{\mathbf{r}_e^{(j)}} = \prod_{j=i-2^\ell}^{i-1} \frac{\tilde{\mathbf{r}}_e^{(j+1)}}{\mathbf{r}_e^{(j)}} \cdot \prod_{j=i-2^\ell+1}^i \frac{\mathbf{r}_e^{(j)}}{\tilde{\mathbf{r}}_e^{(j)}} \geq \prod_{j=i-2^\ell}^{i-1} \frac{\tilde{\mathbf{r}}_e^{(j+1)}}{\mathbf{r}_e^{(j)}} \cdot \exp\left(-\frac{\delta}{10 \log n}\right),$$

So for any $e \in S_\ell$, we have

$$\sum_{j=i-2^\ell}^{i-1} \ln\left(\frac{\tilde{\mathbf{r}}_e^{(j+1)}}{\mathbf{r}_e^{(j)}}\right) \geq \ln\left(\frac{\mathbf{r}_e^{(i)}}{\mathbf{r}_e^{(i-2^\ell)}}\right) - \frac{\delta}{10 \log n} \geq \frac{\delta}{5 \log n}.$$

So we have

$$|S_\ell| \cdot \frac{\delta^2}{(5 \log n)^2} \leq \sum_{e \in S_\ell} \left(\sum_{j=i-2^\ell}^{i-1} \ln\left(\frac{\tilde{\mathbf{r}}_e^{(j+1)}}{\mathbf{r}_e^{(j)}}\right)\right)^2 \leq 2^\ell \cdot \sum_{e \in S_\ell} \sum_{j=i-2^\ell}^{i-1} \ln\left(\frac{\tilde{\mathbf{r}}_e^{(j+1)}}{\mathbf{r}_e^{(j)}}\right)^2 \leq 2^{2\ell} \cdot \zeta,$$

where the last step follows from the first property of the sequence.

So we have $|S_\ell| \leq O(2^{2\ell} (\log n / \delta)^2 \zeta)$. □

B.2 Low Rank Update under ℓ_3 Stability

In this section we prove the low-rank update guarantee under ℓ_3 stability, which holds for MWU with monotone weights, and we only use it in our deterministic algorithm.

B.2.1 Decomposition of Iterations

Lemma B.3 (Decomposition of iterations). *If the weights satisfy that*

$$\sum_{i=1}^T \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \zeta,$$

then we can decompose the T iterations into $\log T + 1$ disjoint sets:

$$B_j := \left\{ i \in [T] \mid \frac{\zeta}{2^{j+1}} < \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \frac{\zeta}{2^j} \right\}, \quad \forall j \in [0 : \log T - 1],$$

$$B_{\log T} := \left\{ i \in [T] \mid \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \frac{\zeta}{T} \right\},$$

and these sets satisfy that $\cup_{j=0}^{\log T} B_j = [T]$, and $|B_j| \leq 2^{j+1}$ for all $j \in [\log T]$.

Proof. Since $\sum_{i=1}^T \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \zeta$, we have that for any $i \in [T]$, $0 \leq \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \zeta$, so each $i \in [T]$ must fall into exactly one set B_j .

For any $j \in [0 : \log T - 1]$, by the definition of B_j we have

$$\sum_{i \in B_j} \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \geq |B_j| \cdot \frac{\zeta}{2^{j+1}}.$$

Combining with our assumption that $\sum_{i=1}^T \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \zeta$, we have that

$$|B_j| \cdot \frac{\zeta}{2^{j+1}} \leq \zeta \Rightarrow |B_j| \leq 2^{j+1}.$$

Finally, note that we trivially have $|B_{\log T}| \leq T < 2^{\log T + 1}$. □

B.2.2 Low Rank Update Scheme under ℓ_3 Stability

Algorithm 4 Low rank update in the t -th iteration

- 1: **procedure** SELECTVECTORL3($\mathbf{r}^{(t)}$)
 - 2: **for** all $j \in [0 : \log T]$ **do**
 - 3: **for** all $\ell \in [0 : \log T]$ **do**
 - 4: **if** $i \in B_j$ and i is the k -th element in B_j where $k \equiv 0 \pmod{2^\ell}$ **then**
 - 5: $I \leftarrow \left\{ e \mid \sum_{k'=k-2^\ell}^k \left| \frac{\mathbf{r}_e^{(B_j[k']+1)}}{\mathbf{r}_e^{(B_j[k'])}} - 1 \right| \geq \frac{\delta}{10 \log^2 n} \right\}$
 - 6: Update the weights for all $e \in I$ to be $\bar{\mathbf{r}}_e^{(t)} \leftarrow \mathbf{r}_e^{(t)}$
-

Lemma B.4 (Low-rank update scheme under ℓ_3 stability). *Assume that the weights are monotonically increasing and satisfy*

$$\sum_{i=1}^T \sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \zeta.$$

Define the sets $B_0, \dots, B_{\log T} \subseteq [T]$ as Lemma B.3, and for any j let $B_j[1], \dots, B_j[|B_j|]$ denote the elements in B_j in increasing order.

For any $\delta \leq 0.1$, Algorithm 4 maintains a vector $\bar{\mathbf{r}} \approx_\delta \mathbf{r}$ where $\bar{\mathbf{r}}$ undergoes the following updates: for any $j \in [0 : \log T]$, for any $\ell \in [0 : \log |B_j|]$, $\bar{\mathbf{r}}$ receives an update of size $O(\zeta \cdot 2^{3\ell-j} \cdot \frac{\log^6 n}{\delta^3})$ in iterations $B_j[2^\ell], B_j[2 \cdot 2^\ell], B_j[3 \cdot 2^\ell], \dots, B_j[\lfloor \frac{|B_j|}{2^\ell} \rfloor \cdot 2^\ell]$.

Proof. For any $j \in [0 : \log T]$, and for any $\ell \in [0 : \log |B_j|]$, in any iteration k that equals to an integer times 2^ℓ , Algorithm 4 performs an update for all coordinates in set I , where

$$I = \left\{ e \mid \sum_{k'=k-2^\ell}^k \left| \frac{\mathbf{r}_e^{(B_j[k']+1)}}{\mathbf{r}_e^{(B_j[k'])}} - 1 \right| \geq \frac{\delta}{10 \log^2 n} \right\}.$$

We first bound the size of the set I . We have

$$\begin{aligned} |I| \cdot \left(\frac{\delta}{10 \log^2 n} \right)^3 &\leq \sum_e \left(\sum_{k'=k-2^\ell}^k \left| \frac{\mathbf{r}_e^{(B_j[k']+1)}}{\mathbf{r}_e^{(B_j[k'])}} - 1 \right| \right)^3 \\ &\leq \sum_e 2^{2\ell} \cdot \sum_{k'=k-2^\ell}^k \left| \frac{\mathbf{r}_e^{(B_j[k']+1)}}{\mathbf{r}_e^{(B_j[k'])}} - 1 \right|^3 \\ &\leq \frac{2^{3\ell} \cdot \zeta}{2^j}, \end{aligned}$$

where the second step follows from $(\sum_{i=1}^n |a_i|)^3 \leq n^2 \cdot \sum_{i=1}^n |a_i|^3$ for any sequence a_i , the third step follows from $\sum_e \left| \frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i)}} - 1 \right|^3 \leq \frac{\zeta}{2^j}$ for all $i \in B_j$.

So we have

$$|I| \leq \zeta \cdot 2^{3\ell-j} \cdot \left(\frac{10 \log^2 n}{\delta} \right)^3.$$

Next we prove that the vector $\bar{\mathbf{r}}$ maintained in Algorithm 4 satisfies $\left| \frac{\bar{\mathbf{r}}_e^{(i)}}{\mathbf{r}_e^{(i)}} - 1 \right| \leq \delta$ for all coordinates e and in all iterations i . Fix a coordinate e and an iteration i , and let i_0 be the last iteration that $\bar{\mathbf{r}}_e$ was updated. For any $j \in [0 : \log T]$, let $B_j[k], B_j[k+1], \dots, B_j[k+t]$ denote the iterations in $[i_0, i]$ that fall into B_j , and note that $t = |[i_0, i] \cap B_j| \leq |B_j| \leq 2^{j+1}$. We can write $k = k_0 < k_1 < k_2 < \dots < k_s = k + t$ where each $k_{\ell+1} - k_\ell$ is a power of 2 and $s \leq 2 \log t \leq 2(j+1)$. Since $\bar{\mathbf{r}}_e$ is not updated in any iterations $B_j[k_1], \dots, B_j[k_s]$, we have that for any $\ell \in [s]$,

$$\sum_{k'=k_{\ell-1}}^{k_\ell} \left| \frac{\mathbf{r}_e^{(B_j[k']+1)}}{\mathbf{r}_e^{(B_j[k'])}} - 1 \right| < \frac{\delta}{10 \log^2 n},$$

so we have

$$\sum_{\ell=1}^s \sum_{k'=k_{\ell-1}}^{k_\ell} \left| \frac{\mathbf{r}_e^{(B_j[k']+1)}}{\mathbf{r}_e^{(B_j[k'])}} - 1 \right| < \frac{s\delta}{10 \log^2 n} \leq \frac{2(j+1)\delta}{10 \log^2 n} \leq \frac{\delta}{\log n}.$$

Since the same argument holds for all $j \in [0 : \log T]$, and each iteration in $[i_0, i]$ falls into exactly one B_j , we have that

$$\sum_{i'=i_0}^{i-1} \left| \frac{\mathbf{r}_e^{(i'+1)}}{\mathbf{r}_e^{(i')}} - 1 \right| < \frac{\delta}{\log n} \cdot \log T \leq \delta.$$

Using the above inequality, and note that the weights are always increasing, we have

$$\begin{aligned} \frac{\mathbf{r}_e^{(i)}}{\overline{\mathbf{r}}_e^{(i)}} &= \frac{\mathbf{r}_e^{(i)}}{\mathbf{r}_e^{(i_0)}} = \prod_{i'=i_0}^{i-1} \left(1 + \left| \frac{\mathbf{r}_e^{(i'+1)}}{\mathbf{r}_e^{(i')}} - 1 \right| \right) \\ &\leq \exp \left(\sum_{i'=i_0}^{i-1} \left| \frac{\mathbf{r}_e^{(i'+1)}}{\mathbf{r}_e^{(i')}} - 1 \right| \right) \leq \exp(\delta). \end{aligned}$$

Finally note that we also have $\frac{\mathbf{r}_e^{(i)}}{\overline{\mathbf{r}}_e} \geq 1$ since the weights are always increasing. \square

Corollary B.5. *For any $j \in [0 : \log T]$ and any t , the total number of coordinates that are updated in iterations $B_j[k], \dots, B_j[k+t]$ is $O\left(\zeta \cdot 2^{3 \log t - j} \cdot \frac{\log^6 n}{\delta^3}\right)$.*

Proof. Using Lemma B.4 we have the total number of updates in iterations $B_j[k], \dots, B_j[k+t]$ is

$$\sum_{\ell=0}^{\log t} O\left(\zeta \cdot 2^{3\ell-j} \cdot \frac{\log^6 n}{\delta^3}\right) \cdot \frac{t}{2^\ell} \leq O\left(\zeta \cdot 2^{3 \log t - j} \cdot \frac{\log^6 n}{\delta^3}\right).$$

\square

C Data Structures

In this section, we would present all the data structures we use for the various tasks in Algorithm 3.

C.1 Inverse Maintenance Data Structure

In this section we present the formal versions of Lemmas 2.2 and 2.3. These are the inverse maintenance data structures of [BNS19], and we have included a version of their results which is tailored to our notations and analysis. For completeness, we include the proofs of the following two lemmas in Section I.

Lemma C.1 (One-level inverse maintenance, Theorem 4.1 of [BNS19]). *There exists a data structure that initially has a matrix $\mathbf{M}^{(0)} \in \mathbb{R}^{n \times n}$, and in each iteration it receives an update $\Delta^{(t)} \in \mathbb{R}^{n \times n}$ to update the matrix to $\mathbf{M}^{(t)} = \mathbf{M}^{(t-1)} + \Delta^{(t)}$. The data structure maintains an iteration counter t_0 and it maintains the inverse $\mathbf{N} = (\mathbf{M}^{(t_0)})^{-1}$ internally. Let $k = \text{nnz}(\Delta^{(t_0+1)}) + \dots + \text{nnz}(\Delta^{(t)})$ denote the total size of the updates until the current iteration. The runtime for each operation of the data structure is as follows:*

- **Initialize**($\mathbf{M}^{(0)}$): Initially set $t_0 = 0$ and $\mathbf{N} = (\mathbf{M}^{(0)})^{-1}$. This operation takes $O(n^\omega)$ time.
- **Update**($\Delta^{(t)}$): The data structure receives the t -th update. This operation takes $O(\text{nnz}(\Delta^{(t)}))$ time.
- **Reset**(t): Reset $t_0 = t$ and $\mathbf{N} = (\mathbf{M}^{(t_0)})^{-1}$. This operation takes $O(\mathcal{T}_{\text{mat}}(n, n, k))$ time.
- **Query**($J_r, J_c \subseteq [n]$): Output the submatrix $((\mathbf{M}^{(t)})^{-1})_{J_r, J_c}$ that has $|J_r| = \ell_r$ rows and $|J_c| = \ell_c$ columns. This operation takes $O(k^\omega + \mathcal{T}_{\text{mat}}(\ell_r, k, \ell_c))$ time.

Lemma C.2 (Two-level inverse maintenance, Theorem 4.2 of [BNS19]). *There exists a data structure that initially has a matrix $\mathbf{M}^{(0)} \in \mathbb{R}^{n \times n}$, and in each iteration it receives an update $\Delta^{(t)} \in \mathbb{R}^{n \times n}$ to update the matrix to $\mathbf{M}^{(t)} = \mathbf{M}^{(t-1)} + \Delta^{(t)}$. The data structure maintains two iteration counters $t_0 \leq t_1$, and it also maintains $k_0 := \text{nnz}(\Delta^{(t_0+1)}) + \dots + \text{nnz}(\Delta^{(t)})$ and $k_1 := \text{nnz}(\Delta^{(t_1+1)}) + \dots + \text{nnz}(\Delta^{(t)})$. Let $J \subseteq [n]$ denote the indexes of the non-zero columns of $\Delta^{(t_0+1)} + \dots + \Delta^{(t_1)}$. For any $t' \leq t$, define the transformation matrix*

$$\mathbf{T}^{(t',t)} := \mathbf{I} + (\mathbf{M}^{(t')})^{-1} \cdot (\mathbf{M}^{(t)} - \mathbf{M}^{(t')}) \in \mathbb{R}^{n \times n}.$$

The data structure maintains $\mathbf{N} = (\mathbf{M}^{(t_0)})^{-1} \in \mathbb{R}^{n \times n}$, and $\mathbf{B} = (\mathbf{T}_{J,J}^{(t_0,t_1)})^{-1}$ that has size at most $k_0 \times k_0$, and $\mathbf{E} = (\mathbf{T}_{J,J}^{(t_0,t_1)})^{-1} \cdot \mathbf{N}_{J,:}$. The runtime for each operation of the data structure is as follows:

- **Initialize**($\mathbf{M}^{(0)}$): *Initially set $t_0 = t_1 = 0$, $\mathbf{B} = 0$, $\mathbf{E} = 0$, and $\mathbf{N} = (\mathbf{M}^{(0)})^{-1}$. This operation takes $O(n^\omega)$ time.*
- **Update**($\Delta^{(t)}$): *The data structure receives the t -th update. This operation takes $O(\text{nnz}(\Delta^{(t)}))$ time.*
- **Reset**(\cdot): *Reset $t_0 = t$ and $\mathbf{N} = (\mathbf{M}^{(t_0)})^{-1}$. This operation takes $O(\mathcal{T}_{\text{mat}}(n, n, k_0))$ time.*
- **PartialReset**(\cdot): *Reset $t_1 = t$, reset $J \subseteq [n]$ to be the indexes of the non-zero columns of $\Delta^{(t_0+1)} + \dots + \Delta^{(t)}$, and reset $\mathbf{B} = (\mathbf{T}_{J,J}^{(t_0,t)})^{-1}$ and $\mathbf{E} = (\mathbf{T}_{J,J}^{(t_0,t)})^{-1} \cdot \mathbf{N}_{J,:}$. This operation takes $O(\mathcal{T}_{\text{mat}}(n, k_0, k_1))$ time.*
- **Query**($J_r, J_c \subseteq [n]$): *Output the submatrix $((\mathbf{M}^{(t)})^{-1})_{J_r, J_c}$ that has $|J_r| = \ell_r$ rows and $|J_c| = \ell_c$ columns. This operation takes $O(\mathcal{T}_{\text{mat}}(k_0, k_1, \max\{k_1, \ell_c\}) + \mathcal{T}_{\text{mat}}(k_0, \ell_r, \ell_c))$ time.*

We can maintain any matrix formula using the inverse maintenance data structure, as shown in [Bra21].

Theorem C.3 (Matrix formula as inverse, Theorem 3.1 of [Bra21]). *Given any formula f with input matrices $\mathbf{A}_1 \in \mathbb{R}^{n_1 \times m_1}, \dots, \mathbf{A}_d \in \mathbb{R}^{n_d \times m_d}$, where the formula f consists of only matrix addition, subtraction, multiplication, and inversion, define $n := \sum_{i=1}^d n_i + m_i$.*

Then there exists a symbolic block matrix \mathbf{N} of size at most $n \times n$, and sets $I, J \subset [n]$, such that for all matrices $\mathbf{A}_1, \dots, \mathbf{A}_d$ for which $f(\mathbf{A}_1, \dots, \mathbf{A}_d)$ is executable, $(\mathbf{N}(\mathbf{A}_1, \dots, \mathbf{A}_d)^{-1})_{I,J} = f(\mathbf{A}_1, \dots, \mathbf{A}_d)$.

Constructing \mathbf{N} from f can be done in $O(n^2)$ time.

C.2 Implicit Inverse Maintenance

In our algorithm, we also require a data structure that allows us to update $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \Delta^{(i,k)}$ in each primal step *implicitly* since we don't have the time budget to query the entire vector $\Delta^{(i,k)}$, and we only query the final vector $\mathbf{x}^{(T)}$ in the end. To solve this problem we present an implicit inverse maintenance data structure, and its proof can be found in Section I. Similar techniques were developed in Section I of [Jia+21] to maintain feasibility.

Lemma C.4 (Implicit two-level inverse maintenance). *There exists a data structure that initially has a matrix $\mathbf{M}^{(0)} \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{v} \in \mathbb{R}^n$, and in each iteration it receives an update $\Delta^{(t)} \in \mathbb{R}^{n \times n}$ to update the matrix to $\mathbf{M}^{(t)} = \mathbf{M}^{(t-1)} + \Delta^{(t)}$. The goal of our algorithm is to support queries that output the sum of inverse vector products $\sum_{i=0}^t (\mathbf{M}^{(i)})^{-1} \cdot \mathbf{v}$ occasionally.*

The data structure maintains two iteration counters $t_0 \leq t_1$. Let $k_0 := \text{nnz}(\Delta^{(t_0+1)}) + \dots + \text{nnz}(\Delta^{(t)})$ and $k_1 := \text{nnz}(\Delta^{(t_1+1)}) + \dots + \text{nnz}(\Delta^{(t)})$. Similar as Lemma C.2, the data structure

maintains $J \subseteq [n]$ that consists of the indexes of the non-zero columns of $\Delta^{(t_0+1)} + \dots + \Delta^{(t_1)}$, $\mathbf{N} = (\mathbf{M}^{(t_0)})^{-1} \in \mathbb{R}^{n \times n}$, $\mathbf{B} = (\mathbf{T}_{J,J}^{(t_0,t_1)})^{-1}$ that has size at most $k_0 \times k_0$, and $\mathbf{E} = (\mathbf{T}_{J,J}^{(t_0,t_1)})^{-1} \cdot \mathbf{N}_{J,:}$. The data structure also maintains three vector $\mathbf{u}_0, \mathbf{u}_1$, and \mathbf{u}_2 that satisfy the invariant:

$$\sum_{i=0}^t (\mathbf{M}^{(i)})^{-1} \mathbf{v} = \mathbf{u}_0 + \mathbf{N} \cdot \mathbf{u}_1 + \begin{bmatrix} \mathbf{N}_{J,:} \cdot \mathbf{u}_2 \\ 0 \end{bmatrix}.$$

The runtime for each operation of the data structure is as follows:

- **Initialize**($\mathbf{M}^{(0)}, \mathbf{v}$): Initially set $t_0 = t_1 = 0$, $\mathbf{B} = 0$, $\mathbf{E} = 0$, $\mathbf{N} = (\mathbf{M}^{(0)})^{-1}$, $\mathbf{u}_0 = (\mathbf{M}^{(0)})^{-1} \mathbf{v}$, and $\mathbf{u}_1 = \mathbf{u}_2 = 0$. This operation takes $O(n^\omega)$ time.
- **Update**($\Delta^{(t)}$): The data structure receives the t -th update and update $\mathbf{u}_0, \mathbf{u}_1$, and \mathbf{u}_2 . This operation takes $O(\mathcal{T}_{\text{mat}}(k_0, k_1, k_1) + n)$ time.
- **Reset**(): Reset $t_0 = t$ and $\mathbf{N} = (\mathbf{M}^{(t_0)})^{-1}$. This operation takes $O(\mathcal{T}_{\text{mat}}(n, n, k_0))$ time.
- **PartialReset**(): Reset $t_1 = t$, reset $J \subseteq [n]$ to be the indexes of the non-zero columns of $\Delta^{(t_0+1)} + \dots + \Delta^{(t)}$, and reset $\mathbf{B} = (\mathbf{T}_{J,J}^{(t_0,t)})^{-1}$ and $\mathbf{E} = (\mathbf{T}_{J,J}^{(t_0,t)})^{-1} \cdot \mathbf{N}_{J,:}$. This operation takes $O(\mathcal{T}_{\text{mat}}(n, k_0, k_1))$ time.
- **QuerySum**(): Output $\sum_{i=0}^t (\mathbf{M}^{(i)})^{-1} \mathbf{v}$. This operation takes $O(n^2)$ time.

C.3 ℓ_3 and ℓ_2 -Norm Estimations

We will also use the following ℓ_3 norm estimation lemma from [WZ13] to estimate the quantity on Line 14 of Algorithm 3.

Lemma C.5 (ℓ_3 norm estimation, Theorem 1 of [WZ13]). *There exists a distribution Π of matrices of size $O(n^{1/3} \log^3 n) \times n$ such that for any vector $\mathbf{x} \in \mathbb{R}^n$, with probability 0.99 we have that a random matrix $\mathbf{U} \sim \Pi$ satisfies*

$$C_3^{-1/3} \|\mathbf{x}\|_3 \leq \|\mathbf{U}\mathbf{x}\|_\infty \leq C_3^{1/3} \|\mathbf{x}\|_3,$$

where $C_3 > 1$ is a constant.

We remark that we can easily boost the success probability of the above theorem to $1 - 1/n^4$ by using $O(\log n)$ copies and take the median of the estimates.

We will also use the standard JL lemma to estimate the Ψ potential which can be written as a ℓ_2 norm.

Lemma C.6 (Johnson-Lindenstrauss Lemma [JL84]). *There exists a function $JL(n, m, \epsilon, \delta)$ that returns a random matrix $\mathbf{J} \in \mathbb{R}^{k \times n}$ where $k = O(\epsilon^{-2} \log(m/\delta))$, and \mathbf{J} satisfies that for any fixed m -element subset $V \subset \mathbb{R}^n$,*

$$\Pr [\forall \mathbf{v} \in V, (1 - \epsilon) \|\mathbf{v}\|_2 \leq \|\mathbf{J}\mathbf{v}\|_2 \leq (1 + \epsilon) \|\mathbf{v}\|_2] \geq 1 - \delta.$$

Furthermore, the function JL runs in $O(kn)$ time.

C.4 ℓ_2 Heavy Hitter

We use a heavy-hitter data structure to get a list of all the large coordinates on which we wish to perform width reduction in Algorithm 3.

Lemma C.7 (ℓ_2 heavy hitter, [Kan+11; Pag13]). *Given any n , ϵ , and δ , there exists a random matrix $\Phi \in \mathbb{R}^{O(\epsilon^{-2} \log(\delta^{-1}) \log n) \times n}$, and a decoding function DECODE , such that given a vector $\mathbf{y} = \Phi \cdot \mathbf{x}$ for some $\mathbf{x} \in \mathbb{R}^n$, $\text{DECODE}(\mathbf{y})$ outputs a list $L \subseteq [n]$ of size $|L| = O(\epsilon^{-2})$, where with probability $1 - \delta$ the list L includes all $i \in [n]$ that satisfies*

$$|\mathbf{x}_i| \geq \epsilon \cdot \|\mathbf{x}\|_2.$$

Furthermore, $\text{DECODE}(\mathbf{y})$ runs in $O(\epsilon^{-2} \log(\delta^{-1}) \log n)$ time.

D Time Complexity of the Randomized Algorithm Using Fast Data Structures

D.1 Implementing MWU Using Fast Data Structures

In this section, we give an algorithm, Algorithm 5, that implements Algorithm 3 using the data structures stated from Section C.

D.2 Correctness of Algorithm

Lemma D.1 (Correctness of Algorithm 5). *The output of Algorithm 5 is the same as that of Algorithm 3.*

Proof. Algorithm 5 implements Algorithm 3 by using the data structures DS_{INV} , $\text{DS}_{\text{IMPLICITINV}}$, DS_{NORM} , $\text{DS}_{\text{HEAVYHITTERS}}$. So it suffices to prove that all these data structures are correct.

Compute $\hat{\mathbf{u}}$ by DS_{INV} . We first prove that on Line 13 of Algorithm 5, the computed vector $\hat{\mathbf{u}}^{(i,k)} \leftarrow (\bar{\mathbf{R}}^{(i,k)})^{-1/2} (\mathbf{S}^{(i)})^\top \cdot \text{DS}_{\text{INV}}.\text{UPDATEQUERY}(\bar{\mathbf{r}}^{(i,k)}, i)$ satisfies

$$\hat{\mathbf{u}}^{(i,k)} = (\bar{\mathbf{R}}^{(i,k)})^{-1/2} \cdot (\mathbf{S}^{(i)})^\top \mathbf{S}^{(i)} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/2} \left(\mathbf{C} (\mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} - \mathbf{I} \right) \mathbf{d},$$

as required by Line 12 of Algorithm 3.

The data structure DS_{INV} (Algorithm 6) uses the two-level inverse maintenance data structure of Lemma C.2 to maintain the inverse of matrix \mathbf{N} that by Lemma C.3 encodes the matrix formula

$$f(\bar{\mathbf{R}}, \mathbf{S}, \mathbf{C}, \mathbf{d}) = \begin{bmatrix} \mathbf{S}^{(0)} \\ \vdots \\ \mathbf{S}^{(T-1)} \end{bmatrix} \cdot \bar{\mathbf{R}}^{1/2} \left(\mathbf{C} (\mathbf{C}^\top \bar{\mathbf{R}} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}} - \mathbf{I} \right) \mathbf{d}.$$

DS_{INV} maintains that its internal variable $\bar{\mathbf{R}} = \bar{\mathbf{R}}^{(i,k)}$ in each iteration, since we update it on Line 7 of Algorithm 6. The output to $\text{DS}_{\text{INV}}.\text{UPDATEQUERY}(\bar{\mathbf{r}}^{(i,k)}, i)$ is (see Line 13 of Algorithm 6)

$$(\mathbf{N}^{-1})_{I_i, J} = \mathbf{S}^{(i)} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/2} \left(\mathbf{C} (\mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} - \mathbf{I} \right) \mathbf{d}.$$

So we have that the $\hat{\mathbf{u}}^{(i,k)}$ is computed as required.

Algorithm 5 Implementing MWU Algorithm Using Fast Data Structures

```

1: procedure MWU-NONMONOTONEROBUST( $\mathbf{C}, \mathbf{d}, \epsilon, a_0, a_1$ )       $\triangleright$  Assume all variables are
   global
2:    $\alpha \leftarrow \tilde{\Theta}(n^{-1/2+\eta}\epsilon)$ 
3:    $\tau \leftarrow \tilde{\Theta}(n^{1/2+\eta}\epsilon^{-4}), \quad \rho \leftarrow \tilde{\Theta}(n^{1/2-3\eta}\epsilon^{-2})$ 
4:    $T \leftarrow \alpha^{-1}\epsilon^{-2} \ln n$ 
5:    $i, k = 0$ 
6:    $b \leftarrow \tilde{\Theta}(n^{1/2+\eta}\epsilon^{-3})$ 
7:    $\mathbf{w}^{(0,0)} \leftarrow \mathbf{1}_n, \quad \mathbf{x}^{(0)} \leftarrow \mathbf{0}_d$ 
8:   Let  $\mathbf{S}^{(0)}, \mathbf{S}^{(1)}, \dots, \mathbf{S}^{(T-1)} \in \mathbb{R}^{b \times n}$  be random matrices as described in Lemma G.8.
9:   Initialize data structures  $\text{DS}_{\text{INV}}, \text{DS}_{\text{NORM}}, \text{DS}_{\text{IMPLICITINV}}, \text{DS}_{\text{HEAVYHITTERS}}$   $\triangleright$  Algorithm 6,
   7, 8, 9
10:  while  $i < T$  do
11:     $\mathbf{r}^{(i,k)} \leftarrow \mathbf{w}^{(i,k)} + \frac{\epsilon}{m} \sum_e \mathbf{w}_e^{(i,k)}$ 
12:     $\bar{\mathbf{r}}^{(i,k)} \leftarrow \text{SELECTVECTOR}(\mathbf{r}^{(i,k)})$ 
13:     $\hat{\mathbf{u}}^{(i,k)} \leftarrow (\mathbf{R}^{(i,k)})^{-1/2} (\mathbf{S}^{(i)})^\top \cdot \text{DS}_{\text{INV}}.\text{UPDATEQUERY}(\bar{\mathbf{r}}^{(i,k)}, i)$ 
14:     $\Psi, \xi \leftarrow \text{DS}_{\text{NORM}}(\bar{\mathbf{r}}^{(i,k)}, i+k) \quad \triangleright \Psi \approx_\epsilon \sum_e \mathbf{r}_e^{(i,k)} (\mathbf{u}_e^{(i,k)})^2, \xi \approx_{C_3} \sum_e \mathbf{r}_e^{(i,k)} |\mathbf{u}_e^{(i,k)}|^3$ 
15:    if  $\xi \leq \rho\Psi$  then
16:       $\vec{\alpha}_e^{(i,k)} = \begin{cases} \alpha \cdot (1 + \epsilon\alpha\hat{\mathbf{u}}_e^{(i,k)}) & \text{if } \hat{\mathbf{u}}_e^{(i,k)} \geq 0 \\ \alpha/(1 - \epsilon\alpha\hat{\mathbf{u}}_e^{(i,k)}) & \text{else} \end{cases}$ 
17:       $\mathbf{w}^{(i+1,k)} \leftarrow \mathbf{w}^{(i,k)} \left(1 + \epsilon \vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)}\right)$ 
18:       $\text{DS}_{\text{IMPLICITINV}}.\text{UPDATE}(\bar{\mathbf{r}}^{(i,k)}) \quad \triangleright$  Implicitly update  $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \Delta^{(i,k)}$ 
19:       $i \leftarrow i + 1$ 
20:    else if  $\xi > \rho\Psi$  then
21:       $L, \mathbf{u}_L^{(i,k)} \leftarrow \text{DS}_{\text{HEAVYHITTERS}}.\text{UPDATEQUERY}(\bar{\mathbf{r}}^{(i,k)})$ 
22:      Let  $S$  be the set of coordinates  $e$  such that  $|\mathbf{u}_e^{(i,k)}| \geq \rho/(2C_3)$ 
23:       $H \subseteq S$  be maximal subset such that  $\sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1}\Psi(\bar{\mathbf{r}}^{(i,k)})$ 
24:      if  $H \neq S$  then
25:        Pick any  $\bar{e} \in S \setminus H$ .
26:        For all  $e \in H \cup \{\bar{e}\}$ ,  $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon)\mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n}\Phi(\mathbf{w}^{(i,k)})$ 
27:         $\mathbf{r}^{(i,k+1)} \leftarrow \mathbf{w}^{(i,k+1)} + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k+1)})$ 
28:        For all  $e \in H \cup \{\bar{e}\}$ ,  $\bar{\mathbf{r}}_e^{(i,k+1)} \leftarrow \mathbf{r}_e^{(i,k+1)}$ 
29:      else
30:        for  $\zeta = \rho, 2\rho, 4\rho, \dots, 2^{c_\rho}\rho$  do
31:           $\triangleright c_\rho$  is defined to be the smallest integer  $c$  that satisfies  $2^c\rho \geq \sqrt{n/\epsilon}$ 
32:          Define the set  $H_\zeta = \{e \in H \mid |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \in [\zeta, 2\zeta]\}$ .
33:          If  $\sum_{e \in H_\zeta} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq \frac{\rho\Psi(\bar{\mathbf{r}}^{(i,k)})}{\log(\frac{n}{\epsilon\rho})}$ , set  $\zeta^* \leftarrow \zeta$ , and break.
34:          For all  $e \in H_{\zeta^*}$ ,  $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon)\mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n}\Phi(\mathbf{w}^{(i,k)})$ 
35:           $\mathbf{r}^{(i,k+1)} \leftarrow \mathbf{w}^{(i,k+1)} + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k+1)})$ 
36:          For all  $e \in H_{\zeta^*}$ ,  $\bar{\mathbf{r}}_e^{(i,k+1)} \leftarrow \mathbf{r}_e^{(i,k+1)}$ 
37:         $k \leftarrow k + 1$ 
38:       $\mathbf{x}^{(T)} \leftarrow \text{DS}_{\text{IMPLICITINV}}.\text{QUERY}()$ 
39:    return  $\mathbf{x}^{(T)}/T$ 

```

Algorithm 6 Inverse maintenance data structure DS_{INV} to compute $\hat{\mathbf{u}}$

- 1: **procedure** INITIALIZE(\cdot)
- 2: $\mathbf{S} \leftarrow [(\mathbf{S}^{(0)})^\top, (\mathbf{S}^{(1)})^\top, \dots, (\mathbf{S}^{(T-1)})^\top]^\top \in \mathbb{R}^{bT \times n}$
- 3: $\bar{\mathbf{r}} \leftarrow \mathbf{r}^{(0)}$
- 4: Let \mathbf{N} be the matrix given by Lemma C.3 that encodes the matrix formula

$$f(\mathbf{R}, \mathbf{S}, \mathbf{C}, \mathbf{d}) = \mathbf{S} \cdot \mathbf{R}^{1/2} \left(\mathbf{C}(\mathbf{C}^\top \mathbf{R} \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{R} - \mathbf{I} \right) \mathbf{d},$$

i.e., there exist index sets I, J such that $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{R}, \mathbf{S}, \mathbf{C}, \mathbf{d})$. Also let $I_0, I_1, \dots, I_{T-1} \subset I$ each of size b , denote the indexes of the rows corresponding to $\mathbf{S}^{(0)}, \mathbf{S}^{(1)}, \dots, \mathbf{S}^{(T-1)}$.

- 5: $\text{DS}.\text{INITIALIZE}(\mathbf{N})$ where DS is the two-level inverse maintenance data structure of Lemma C.2.
 - 6: **procedure** UPDATEQUERY($\bar{\mathbf{r}}^{\text{new}}, i$)
 - 7: $\text{DS}.\text{UPDATE}(\Delta)$, where $\Delta = \bar{\mathbf{R}}^{\text{new}} - \bar{\mathbf{R}}$
 - 8: $\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}^{\text{new}}$
 - 9: **if** $\text{DS}.k_0 \geq n^{a_0}$ **then**
 - 10: $\text{DS}.\text{RESET}()$
 - 11: **else if** $\text{DS}.k_1 \geq n^{a_1}$ **then**
 - 12: $\text{DS}.\text{PARTIALRESET}()$
 - 13: **return** $\text{DS}.\text{QUERY}(I_i, J)$ $\triangleright |I_i| = b$ and $|J| = 1$
-

Compute \mathbf{x} by $\text{DS}_{\text{IMPLICITINV}}$. Next we prove that Line 18 of Algorithm 5 implicitly updates $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \Delta^{(i,k)}$, as required by Line 18 of Algorithm 3, and that Line 38 of Algorithm 5 outputs the correct $\mathbf{x}^{(T)}$.

The data structure $\text{DS}_{\text{IMPLICITINV}}$ (Algorithm 8) uses the implicit inverse maintenance data structure of Lemma C.4 to maintain the inverse of matrix \mathbf{N} that by Lemma C.3 encodes the matrix formula

$$f(\bar{\mathbf{R}}, \mathbf{C}) = (\mathbf{C}^\top \bar{\mathbf{R}} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}.$$

We also initialize this data structure with the vector \mathbf{d}' (Line 4 of Algorithm 8), and by Lemma C.4 the algorithm maintains

$$\begin{aligned} \sum_{i=0}^{(T-1)} \left((\mathbf{N}^{(i)})^{-1} \cdot \mathbf{d}' \right)_{I_i} &= \sum_{i=0}^{(T-1)} f(\bar{\mathbf{R}}^{(i,k)}, \mathbf{C}) \cdot \mathbf{d} \\ &= \sum_{i=0}^{(T-1)} (\mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \cdot \mathbf{d}, \end{aligned}$$

and this is exactly $\mathbf{x}^{(T)} = \sum_{i=0}^{(T-1)} \Delta^{(i,k)}$ where $\Delta^{(i,k)} = (\mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{d}$ is what we need to compute (see Line 10 of Algorithm 3).

Compute norms by DS_{NORM} . Next we show that Line 14 of Algorithm 5 computes

$$\Psi \approx_\epsilon \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\mathbf{u}_e^{(i,k)})^2, \text{ and } \xi \approx_{C_3} \sum_e \bar{\mathbf{r}}_e^{(i,k)} |\mathbf{u}_e^{(i,k)}|^3.$$

Similar to the proof for DS_{INV} , in each iteration DS_{NORM} (see Algorithm 7) maintains

$$\begin{aligned} &\mathbf{J} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/2} (\mathbf{C}(\mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} - \mathbf{I}) \mathbf{d}, \\ \text{and } &\mathbf{U} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/3} (\mathbf{C}(\mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \bar{\mathbf{R}}^{(i,k)} - \mathbf{I}) \mathbf{d}. \end{aligned}$$

Algorithm 7 Data structures DS_{NORM} to approximately compute ℓ_2 and ℓ_3 norms

- 1: **procedure** INITIALIZE()
- 2: Let $\mathbf{J}^{(0)}, \dots, \mathbf{J}^{(T+K)} \in \mathbb{R}^{O(\epsilon^{-2} \log(n)) \times n}$ be random JL matrices as described in Lemma C.6.
- 3: $\mathbf{J} \leftarrow [(\mathbf{J}^{(0)})^\top, (\mathbf{J}^{(1)})^\top, \dots, (\mathbf{J}^{(T+K)})^\top]^\top \in \mathbb{R}^{O(\epsilon^{-2} \log(n)(T+K)) \times n}$
- 4: Let $\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(T+K)} \in \mathbb{R}^{O(n^{1/3} \log^3(n)) \times n}$ be random matrices as described in Lemma C.5.
- 5: $\mathbf{U} \leftarrow [(\mathbf{U}^{(0)})^\top, (\mathbf{U}^{(1)})^\top, \dots, (\mathbf{U}^{(T+K)})^\top]^\top \in \mathbb{R}^{O(n^{1/3} \log^3(n)(T+K)) \times n}$
- 6: $\bar{\mathbf{r}} \leftarrow \mathbf{r}^{(0)}$
- 7: Let \mathbf{N}_{ℓ_2} and \mathbf{N}_{ℓ_3} be the matrices given by Lemma C.3 that encodes the matrix formulas

$$f_{\ell_2}(\mathbf{R}, \mathbf{J}, \mathbf{C}, \mathbf{d}) = \mathbf{J} \mathbf{R}^{1/2} (\mathbf{C} (\mathbf{C}^\top \mathbf{R} \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{R} - \mathbf{I}) \mathbf{d},$$

$$f_{\ell_3}(\mathbf{R}, \mathbf{U}, \mathbf{C}, \mathbf{d}) = \mathbf{U} \mathbf{R}^{1/3} (\mathbf{C} (\mathbf{C}^\top \mathbf{R} \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{R} - \mathbf{I}) \mathbf{d},$$

i.e., there exist index sets $I_{\ell_2}, J_{\ell_2}, I_{\ell_3}, J_{\ell_3}$ such that $(\mathbf{N}_{\ell_2}^{-1})_{I_{\ell_2}, J_{\ell_2}} = f_{\ell_2}(\mathbf{R}, \mathbf{J}, \mathbf{C}, \mathbf{d})$ and $(\mathbf{N}_{\ell_3}^{-1})_{I_{\ell_3}, J_{\ell_3}} = f_{\ell_3}(\mathbf{R}, \mathbf{U}, \mathbf{C}, \mathbf{d})$. Also let $I_{\ell_2,0}, \dots, I_{\ell_2,T+K} \subset I_{\ell_2}$ denote the index sets of the rows corresponding to $\mathbf{J}^{(0)}, \dots, \mathbf{J}^{(T+K)}$, and let $I_{\ell_3,0}, \dots, I_{\ell_3,T+K} \subset I_{\ell_3}$ denote the rows corresponding to $\mathbf{U}^{(0)}, \dots, \mathbf{U}^{(T+K)}$.

- 8: $\text{DS}_{\ell_2}.\text{INITIALIZE}(\mathbf{N}_{\ell_2})$ and $\text{DS}_{\ell_3}.\text{INITIALIZE}(\mathbf{N}_{\ell_3})$ by Lemma C.2.
 - 9: **procedure** UPDATEQUERY($\bar{\mathbf{r}}^{\text{new}}, i$)
 - 10: $\text{DS}_{\ell_2}.\text{UPDATE}(\Delta)$ and $\text{DS}_{\ell_3}.\text{UPDATE}(\Delta)$, where $\Delta = \bar{\mathbf{R}}^{\text{new}} - \bar{\mathbf{R}}$
 - 11: $\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}^{\text{new}}$
 - 12: **if** $\text{DS}_{\ell_2}.k_0 \geq n^{a_0}$ **then**
 - 13: $\text{DS}_{\ell_2}.\text{RESET}()$ and $\text{DS}_{\ell_3}.\text{RESET}()$
 - 14: **else if** $\text{DS}_{\ell_2}.k_1 \geq n^{a_1}$ **then**
 - 15: $\text{DS}_{\ell_2}.\text{PARTIALRESET}()$ and $\text{DS}_{\ell_3}.\text{PARTIALRESET}()$
 - 16: **return** $(\|\text{DS}_{\ell_2}.\text{QUERY}(I_{\ell_2,i}, J_{\ell_2})\|_2^2, \|\text{DS}_{\ell_3}.\text{QUERY}(I_{\ell_3,i}, J_{\ell_3})\|_\infty^3)$
-

Algorithm 8 Implicit inverse maintenance data structure $\text{DS}_{\text{IMPLICITINV}}$ to compute Δ and to update \mathbf{x}

- 1: **procedure** INITIALIZE()
 - 2: $\bar{\mathbf{r}} \leftarrow \mathbf{r}^{(0)}$
 - 3: Let \mathbf{N} be the matrix given by Lemma C.3 that encodes the matrix formula $f(\mathbf{R}, \mathbf{C}) = (\mathbf{C}^\top \mathbf{R} \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{R}$, i.e., there exist index sets I, J such that $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{R}, \mathbf{C})$.
 - 4: $\text{DS}.\text{INITIALIZE}(\mathbf{N}, \mathbf{d}')$ where DS is the implicit inverse maintenance data structure of Lemma C.4, and \mathbf{d}' has the same size as \mathbf{N} , and it equals to \mathbf{d} in J , and its other coordinates are all zero.
 - 5: **procedure** UPDATE($\bar{\mathbf{r}}^{\text{new}}$)
 - 6: $\text{DS}.\text{UPDATE}(\Delta)$, where $\Delta = \bar{\mathbf{R}}^{\text{new}} - \bar{\mathbf{R}}$
 - 7: $\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}^{\text{new}}$
 - 8: **if** $\text{DS}.k_0 \geq n^{a_0}$ **then**
 - 9: $\text{DS}.\text{RESET}()$
 - 10: **else if** $\text{DS}.k_1 \geq n^{a_1}$ **then**
 - 11: $\text{DS}.\text{PARTIALRESET}()$
 - 12: **procedure** QUERYSUM()
 - 13: **return** $\text{DS}.\text{QUERYSUM}()$
-

Algorithm 9 Heavy hitter data structure $\text{DS}_{\text{HEAVYHITTERS}}$ to compute the heavy entries of \mathbf{u}

- 1: **procedure** INITIALIZE()
- 2: $\epsilon_{\text{heavy}} \leftarrow \frac{\rho\sqrt{\epsilon}}{2C_3\sqrt{n}}$
- 3: $\bar{\mathbf{r}} \leftarrow \mathbf{r}^{(0)}$
- 4: Let $\Phi \in \mathbb{R}^{O(\epsilon_{\text{heavy}}^{-2} \log^2 n) \times n}$ be the random matrix as described in Lemma C.7.
- 5: Let \mathbf{N} and \mathbf{N}_Φ be the matrix given by Lemma C.3 that encodes the matrix formulas

$$f(\mathbf{R}, \mathbf{C}, \mathbf{d}) = \mathbf{R}^{1/2}(\mathbf{C}(\mathbf{C}^\top \mathbf{R} \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{R} - \mathbf{I})\mathbf{d},$$

$$f_\Phi(\Phi, \mathbf{R}, \mathbf{C}, \mathbf{d}) = \Phi \cdot \mathbf{R}^{1/2}(\mathbf{C}(\mathbf{C}^\top \mathbf{R} \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{R} - \mathbf{I})\mathbf{d},$$

i.e., there exist index sets I, J, I_Φ, J_Φ such that $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{R}, \mathbf{C}, \mathbf{d})$ and $(\mathbf{N}_\Phi^{-1})_{I_\Phi, J_\Phi} = f_\Phi(\Phi, \mathbf{R}, \mathbf{C}, \mathbf{d})$.

- 6: $\text{DS}.\text{INITIALIZE}(\mathbf{N})$, $\text{DS}_\Phi.\text{INITIALIZE}(\mathbf{N}_\Phi)$, where DS and DS_Φ are both the inverse maintenance data structure of Lemma C.2.
 - 7: **procedure** UPDATEQUERY($\bar{\mathbf{r}}^{\text{new}}$)
 - 8: $\text{DS}.\text{UPDATE}(\Delta)$ and $\text{DS}_\Phi.\text{UPDATE}(\Delta)$, where $\Delta = \bar{\mathbf{R}}^{\text{new}} - \bar{\mathbf{R}}$
 - 9: $\bar{\mathbf{r}} \leftarrow \bar{\mathbf{r}}^{\text{new}}$
 - 10: **if** $\text{DS}.k_0 \geq n^{a_0}$ **then**
 - 11: $\text{DS}.\text{RESET}()$ and $\text{DS}_\Phi.\text{RESET}()$
 - 12: **else if** $\text{DS}.k_1 \geq n^{a_1}$ **then**
 - 13: $\text{DS}.\text{PARTIALRESET}()$ and $\text{DS}_\Phi.\text{PARTIALRESET}()$
 - 14: $\mathbf{y} \leftarrow \text{DS}_\Phi.\text{QUERY}(I_\Phi, J_\Phi)$
 - 15: $L \leftarrow \text{DECODE}(\mathbf{y})$, where $\text{DECODE}()$ is the decoding algorithm of Lemma C.7. We can view $L \subset [n]$ as a subset of I .
 - 16: **return** $(L, \text{DS}.\text{QUERY}(L, J))$
-

And on Line 16 of Algorithm 7 it outputs

$$\Psi = \|\mathbf{J}^{(i)} \cdot (\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}\|_2^2, \text{ and } \xi = \|\mathbf{U}^{(i)} \cdot (\overline{\mathbf{R}}^{(i,k)})^{1/3} \mathbf{u}^{(i,k)}\|_\infty^3,$$

where $\mathbf{u}^{(i,k)} = (\mathbf{C}(\mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} - \mathbf{I})\mathbf{d}$ is as required by Line 11 of Algorithm 3.

Then by Lemma C.6 and C.5, and since we use a new random matrix $\mathbf{J}^{(i)}$ and $\mathbf{U}^{(i)}$ in each iteration, we have that with probability $1 - 1/n^e$, for all iterations we have

$$(1 - \epsilon) \|(\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}\|_2^2 \leq \Psi \leq (1 + \epsilon) \|(\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}\|_2^2, \\ C_3^{-1} \|(\overline{\mathbf{R}}^{(i,k)})^{1/3} \mathbf{u}^{(i,k)}\|_3^3 \leq \xi \leq C_3 \|(\overline{\mathbf{R}}^{(i,k)})^{1/3} \mathbf{u}^{(i,k)}\|_3^3.$$

Compute set S for the width reduction step by $\text{DS}_{\text{HEAVYHITTERS}}$. Finally we prove that Line 21 of Algorithm 5 computes the set $S = \{e : |\mathbf{u}_e^{(i,k)}| \geq \rho/(2C_3)\}$ and the values of $\mathbf{u}_e^{(i,k)}$ for all $e \in S$, as by Line 21 of Algorithm 3.

Similar to the proof for DS_{INV} , in each iteration $\text{DS}_{\text{HEAVYHITTERS}}$ (see Algorithm 9) maintains

$$\Phi \cdot (\overline{\mathbf{R}}^{(i,k)})^{1/2} (\mathbf{C}(\mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} - \mathbf{I})\mathbf{d}, \\ \text{and } (\overline{\mathbf{R}}^{(i,k)})^{1/2} (\mathbf{C}(\mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} - \mathbf{I})\mathbf{d}.$$

And on Line 14 of Algorithm 9 it computes

$$\mathbf{y} = \Phi \cdot (\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)},$$

where $\mathbf{u}^{(i,k)} = (\mathbf{C}(\mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{C})^{-1} \mathbf{C}^\top \overline{\mathbf{R}}^{(i,k)} - \mathbf{I})\mathbf{d}$ is as required by Line 11 of Algorithm 3.

Then on Line 15 of Algorithm 9 it decodes \mathbf{y} and compute the set L , and by Lemma C.7, with probability $1 - 1/n^4$, L includes all $e \in [n]$ that satisfies

$$|(\overline{\mathbf{r}}^{(i,k)})_e^{1/2} \mathbf{u}_e^{(i,k)}| \geq \epsilon_{\text{heavy}} \cdot \|(\overline{\mathbf{r}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}\|_2.$$

Note that for any $e \in S$, we have

$$|\mathbf{u}_e^{(i,k)}| \geq \rho/(2C_3) \Rightarrow \mathbf{r}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2 \geq \frac{\rho^2}{4C_3^2} \cdot \mathbf{r}_e^{(i,k)} \\ \Rightarrow \mathbf{r}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2 \geq \frac{\rho^2 \epsilon}{4C_3^2 n} \cdot \Psi(\mathbf{r}^{(i,k)}) \\ \Rightarrow \mathbf{r}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2 \geq \epsilon_{\text{heavy}}^2 \cdot \|(\overline{\mathbf{r}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}\|_2^2,$$

where the second step follows from $\mathbf{r}_e^{(i,k)} \geq \frac{\epsilon}{n} \Psi(\mathbf{r}^{(i,k)})$, and the third follows from $\epsilon_{\text{heavy}} = \frac{\rho\sqrt{\epsilon}}{2C_3\sqrt{n}}$. This means we have

$$S \subseteq L,$$

and it suffices to enumerate all $e \in L$ to check if $|\mathbf{u}_e^{(i,k)}| \geq \rho/(2C_3)$ and compute S .

In the output on Line 16 of Algorithm 9, we also output $\text{QUERY}(L, J)$ which computes $\mathbf{u}_L^{(i,k)}$ exactly.

Finally, note that we can re-use the random matrix Φ because the set S and $\mathbf{u}_e^{(i,k)}$ for $e \in S$ are computed exactly, so the next iteration does not depend on the randomness of Φ . \square

D.3 Time Complexity under ℓ_2 Stability

In this section we bound the time complexity of Algorithm 5.

Theorem D.2 (Time complexity of Algorithm 5). *For any parameters that satisfy $a_0 \leq \alpha_*$ and $a_1 \leq a_0 \cdot \alpha_*$, the time complexity of Algorithm 5 is*

$$\tilde{O}\left(n^\omega + n^{2.5-a_0/2} + n^{1.5+a_0-a_1/2} + n^{1/2-\eta+a_0+(\omega-1)a_1}\right) \cdot \text{poly}(\epsilon^{-1}).$$

In particular, when $\omega = 2 + o(1)$, this time complexity is bounded by

$$\tilde{O}\left(n^{2+1/22.5}\right) \text{poly}(\epsilon^{-1}).$$

Proof. The dominating steps of Algorithm 5 are the operations involving the data structures, since all other operations can be computed in $O(n \log n)$ time per iteration. We will focus on bounding the runtimes of the data structure operations.

Initialization The INITIALIZE operations of the four data structures DS_{INV} , $\text{DS}_{\text{IMPLICITINV}}$, DS_{NORM} , $\text{DS}_{\text{HEAVYHITTERS}}$ are each called once on Line 9 of Algorithm 5.

- DS_{INV} : By Lemma C.3, the matrix \mathbf{N} of Algorithm 6 has size $N \times N$ where

$$N = \max\{bT, n\} = \tilde{\Theta}(n\epsilon^{-5} \ln^5 n),$$

since $T = \alpha^{-1}\epsilon^{-2} \ln n$, $b = \frac{n\alpha \log^4 n}{\epsilon^3}$. By Lemma C.2, the initialization time of the two level inverse maintenance data structure is

$$O(N^\omega) = \tilde{\Theta}(n^\omega \epsilon^{-5\omega}).$$

- $\text{DS}_{\text{IMPLICITINV}}$: By Lemma C.3, the matrix \mathbf{N} of Algorithm 8 has size $\tilde{O}_\epsilon(n) \times \tilde{O}_\epsilon(n)$. By Lemma C.4, the initialization time of the implicit inverse maintenance data structure is $\tilde{O}_\epsilon(n^\omega)$.
- DS_{NORM} : By Lemma C.3, the matrices \mathbf{N}_{ℓ_2} and \mathbf{N}_{ℓ_3} of Algorithm 7 have size $N_{\ell_2} \times N_{\ell_2}$ and $N_{\ell_3} \times N_{\ell_3}$, where

$$\begin{aligned} N_{\ell_2} &= \max\{O(\epsilon^{-2} \log(n)(T + K)), n\} = \tilde{\Theta}_\epsilon(\max\{n^{1/2-\eta}, n\}), \\ N_{\ell_3} &= \max\{O(n^{1/3} \log^3(n)(T + K)), n\} = \tilde{\Theta}_\epsilon(\max\{n^{5/6-\eta}, n\}). \end{aligned}$$

From Lemma C.2, the initialization time of the two inverse maintenance data structures is

$$O(N_{\ell_2}^\omega + N_{\ell_3}^\omega) = \tilde{\Theta}_\epsilon(\max\{n^{(1/2-\eta)\omega}, n^{(5/6-\eta)\omega}, n^\omega\}).$$

- $\text{DS}_{\text{HEAVYHITTERS}}$: By Lemma C.3, the matrix \mathbf{N} of Algorithm 7 has size $O(n) \times O(n)$, and \mathbf{N}_Φ has size $N_\Phi \times N_\Phi$, where

$$N_\Phi = \max\{O(\epsilon_{\text{heavy}}^{-2} \log^2 n), n\} = \tilde{\Theta}_\epsilon(\max\{n^{6\eta}, n\})$$

since $\epsilon_{\text{heavy}} = \frac{\rho\sqrt{\epsilon}}{2C_3\sqrt{n}}$ and $\rho = \tilde{\Theta}(n^{1/2-3\eta}\epsilon^{-2})$. By Lemma C.2, the initialization time of the two inverse maintenance data structures is

$$O(n^\omega + N_\Phi^\omega) = \tilde{\Theta}_\epsilon(\max\{n^{6\eta\omega}, n^\omega\}).$$

Summing up all these terms, and since we set $\eta = 1/10$, the total initialization time is

$$\tilde{O}_\epsilon(n^\omega).$$

Reset Since Algorithm 5 implements Algorithm 3, it satisfies the low-rank update scheme of Theorem 3.3, so we have that the sequence $\bar{\mathbf{r}}^{(i,k)}$ undergoes at most $\frac{T+K}{2^\ell}$ number of updates of size

$$\tilde{O}_\epsilon \left(\left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot 2^{2\ell} \right) \quad (5)$$

for every $\ell \in [0 : \log T]$.

Note that the four data structures DS_{INV} , $\text{DS}_{\text{IMPLICITINV}}$, DS_{NORM} , $\text{DS}_{\text{HEAVYHITTERS}}$ all follow the same reset and partial reset scheme. And since the matrix maintained by DS_{INV} has the largest size $N = \tilde{\Theta}(n\epsilon^{-5})$, it suffices to bound the reset and partial reset time of DS_{INV} .

As stated on Line 10 of Algorithm 6, we only perform the RESET operation in the i -th iteration if $k_0 \geq n^{a_0}$, where $k_0 = \text{nnz}(\bar{\mathbf{r}}^{(i,k)} - \bar{\mathbf{r}}_0)$, and $\bar{\mathbf{r}}_0$ is the variable maintained by the data structure that was updated in the last RESET (see Lemma C.2). By the low rank update size of Eq. (5), for any $a \in [a_0, 1]$, we only accumulate updates of size n^a for at most $\frac{T+K}{n^{a/2-\eta}} \cdot \frac{\log n}{\delta}$ number of times. For convenience for any a we define a parameter $\ell = \log(n^{a/2-\eta} \cdot \frac{\delta}{\log n})$ such that we get a update of size $n^a = \left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot 2^{2\ell}$ for at most $(T+K)/2^\ell$ times. Let $\ell_0 = \log(n^{a_0/2-\eta} \cdot \frac{\delta}{\log n})$, and by Lemma C.2 the total reset time over all iterations is

$$\tilde{O}_\epsilon \left(\sum_{\ell=\ell_0}^{\log T} \frac{T+K}{2^\ell} \cdot \mathcal{T}_{\text{mat}} \left(N, N, \left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot 2^{2\ell} \right) \right).$$

Define $x(\ell) = \log_N((\frac{\log n}{\delta})^2 \cdot n^{2\eta} \cdot 2^{2\ell}) = \log_N((\frac{\log n}{\delta})^2 \cdot n^{2\eta}) + \frac{2\ell}{\log N}$, then since we defined $\omega(x)$ such that $\mathcal{T}_{\text{mat}}(n, n, n^x) = n^{\omega(x)+o(1)}$, we have the above time complexity equals to

$$\tilde{O}_\epsilon \left((T+K) \cdot \sum_{\ell=\ell_0}^{\log T} N^{\omega(x(\ell)) - \frac{\ell}{\log N}} \right).$$

By Fact A.6 we know that $\omega(x)$ is convex, and hence the function $f(\ell) = \omega(x(\ell)) - \frac{\ell}{\log N}$ is also convex. So we have that the summation is upper bounded by the terms $\ell = \ell_0$ and $\ell = \log T$. And so the above time complexity is bounded by

$$\begin{aligned} & \tilde{O}_\epsilon \left(\frac{T+K}{2^{\ell_0}} \cdot \mathcal{T}_{\text{mat}} \left(N, N, \left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot 2^{2\ell_0} \right) + \mathcal{T}_{\text{mat}} \left(N, N, \left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot T^2 \right) \right) \\ & \leq \tilde{O} \left(n^{1/2-a_0/2} \cdot \mathcal{T}_{\text{mat}}(n, n, n^{a_0}) + \mathcal{T}_{\text{mat}}(n, n, n) \right) \cdot \text{poly}(\epsilon^{-1}), \end{aligned}$$

since $\ell_0 = \log(n^{a_0/2-\eta} \cdot \frac{\delta}{\log n})$, $N = \tilde{\Theta}_\epsilon(n)$, $T = \alpha^{-1}\epsilon^{-2} \ln n$, $\alpha = \tilde{\Theta}_\epsilon(n^{-1/2+\eta})$.

Partial reset Similar to the RESET operation, it suffices to bound the partial reset time of DS_{INV} . As stated on Line 12 of Algorithm 6, we only perform the PARTIALRESET operation in the i -th iteration if $k_1 \geq n^{a_1}$, where $k_1 = \text{nnz}(\bar{\mathbf{r}}^{(i,k)} - \bar{\mathbf{r}}_1)$, and $\bar{\mathbf{r}}_1$ is the variable maintained by the data structure that was updated in the last PARTIALRESET (see Lemma C.2). Let $\ell_1 = \log(n^{a_1/2-\eta} \cdot \frac{\delta}{\log n})$ such that we perform a PARTIALRESET of size n^{a_1} for at most $\frac{T+K}{2^{\ell_1}}$ times, and by Lemma C.2 the total partial reset time over all iterations is

$$\tilde{O}_\epsilon \left(\sum_{\ell=\ell_1}^{\log T} \frac{T+K}{2^\ell} \cdot \mathcal{T}_{\text{mat}} \left(N, n^{a_0}, \left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot 2^{2\ell} \right) \right).$$

Using the same argument as for reset operation and using the convexity of $\omega_{a_0}(x)$ (Fact A.6), we have that the summation is upper bounded by the terms $\ell = \ell_1$ and $\ell = \log T$. And so the above time complexity is bounded by

$$\begin{aligned} & \tilde{O}_\epsilon \left(\frac{T+K}{2^{\ell_1}} \cdot \mathcal{T}_{\text{mat}} \left(N, n^{a_0}, \left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot 2^{2\ell_1} \right) + \mathcal{T}_{\text{mat}} \left(N, n^{a_0}, \left(\frac{\log n}{\delta} \right)^2 \cdot n^{2\eta} \cdot T^2 \right) \right) \\ & \leq \tilde{O} \left(n^{1/2-a_1/2} \cdot \mathcal{T}_{\text{mat}}(n, n^{a_0}, n^{a_1}) + \mathcal{T}_{\text{mat}}(n, n^{a_0}, n) \right) \cdot \text{poly}(\epsilon^{-1}) \end{aligned}$$

since $\ell_1 = \log(n^{a_1/2-\eta} \cdot \frac{\delta}{\log n})$, $N = \tilde{\Theta}_\epsilon(n)$, $T = \alpha^{-1}\epsilon^{-2} \ln n$, $\alpha = \tilde{\Theta}_\epsilon(n^{-1/2+\eta})$.

Query. Next we bound the runtime of the query operations of the four data structures.

- **DS_{INV}:** In DS_{INV} (Algorithm 6), the query operation is called with sets $|I_i| = b$ and $|J| = 1$, and by Lemma C.2 its runtime per iteration is

$$O(\mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) + n^{a_0} \cdot b) = \tilde{O}_\epsilon(\mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) + n^{a_0+1/2+\eta} \cdot)$$

since $b = \tilde{\Theta}_\epsilon(n^{1/2+\eta})$.

- **DS_{IMPLICITINV}:** In Algorithm 5, the QUERY_{SUM} operation of DS_{IMPLICITINV} (Algorithm 8) is only called once in the algorithm, and by Lemma C.4 its runtime over all iterations is $O(n^2)$.
- **DS_{NORM}:** In DS_{NORM} (Algorithm 7), the query operation is called with sets $|I_{\ell_2, i}| = O(\epsilon^{-2} \log(n))$ and $|J_{\ell_2}| = 1$, and with sets $|I_{\ell_3, i}| = O(n^{1/3} \log^3(n))$ and $|J_{\ell_3}| = 1$, and by Lemma C.2 its runtime per iteration is

$$O(\mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) + n^{a_0} \cdot (\epsilon^{-2} \log(n) + n^{1/3} \log^3(n))).$$

- **DS_{HEAVYHITTERS}:** In DS_{HEAVYHITTERS} (Algorithm 9), the query operation is called with sets $|I_\Phi| = O(\epsilon_{\text{heavy}}^{-2} \log^2 n)$ and $|J_\Phi| = 1$, and with sets $|L| = O(\epsilon_{\text{heavy}}^{-2})$ and $|J| = 1$, and by Lemma C.2 its runtime per iteration is

$$O(\mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) + n^{a_0} \cdot \epsilon_{\text{heavy}}^{-2} \log^2 n) = \tilde{O}_\epsilon(\mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) + n^{a_0+6\eta} \cdot),$$

since $\epsilon_{\text{heavy}} = \frac{\rho\sqrt{\epsilon}}{2C_3\sqrt{n}}$ and $\rho = \tilde{\Theta}_\epsilon(n^{1/2-3\eta})$.

Combining these four query time, we have that over all iterations, the total query time is

$$\begin{aligned} & \tilde{O} \left(T \cdot \left(\mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) + n^{a_0+1/2+\eta} + n^{a_0+1/3} + n^{a_0+6\eta} \right) \right) \cdot \text{poly}(\epsilon^{-1}) \\ & = \tilde{O} \left(n^{1/2-\eta} \cdot \mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) + n^{1+a_0} \right) \cdot \text{poly}(\epsilon^{-1}), \end{aligned}$$

since $T = \alpha^{-1}\epsilon^{-2} \ln n$, $\alpha = \tilde{\Theta}_\epsilon(n^{-1/2+\eta})$, and we set $\eta = 1/10$.

Total runtime. Combining the time complexities of initialization, reset, partial reset, and query, we have that the total time complexity is bounded by $\text{poly}(\epsilon^{-1})$ times

$$\tilde{O} \left(n^\omega + n^{1/2-a_0/2} \cdot \mathcal{T}_{\text{mat}}(n, n, n^{a_0}) + n^{1/2-a_1/2} \cdot \mathcal{T}_{\text{mat}}(n, n^{a_0}, n^{a_1}) + n^{1/2-\eta} \cdot \mathcal{T}_{\text{mat}}(n^{a_0}, n^{a_1}, n^{a_1}) \right).$$

When $a_0 \leq \alpha_*$ and $a_1 \leq a_0 \cdot \alpha_*$, this becomes

$$\tilde{O} \left(n^\omega + n^{2.5-a_0/2} + n^{1.5+a_0-a_1/2} + n^{1/2-\eta+a_0+(\omega-1)a_1} \right) \cdot \text{poly}(\epsilon^{-1}).$$

□

Time complexity when $\omega = 2$. We remark that when $\omega = 2$, we can choose the optimal trade-off $a_0 = 1 - \frac{1-2\eta}{9}$ and $a_1 = 1 - \frac{1-2\eta}{3}$, and the time complexity becomes

$$\tilde{O}\left(n^{2+(1-2\eta)/18}\right) \cdot \text{poly}(\epsilon^{-1}).$$

Since we choose $\eta = 1/10$, this become

$$\tilde{O}\left(n^{2+1/22.5}\right) \cdot \text{poly}(\epsilon^{-1}).$$

E Time Complexity of the Deterministic Algorithm Using Fast Data Structures

In this section we show that we can implement Algorithm 1 by using the one-level inverse maintenance data structure (Lemma C.1). This algorithm is deterministic since we don't use any randomized techniques.

Theorem E.1 (Time complexity of Algorithm 1 using one-level inverse maintenance). *For any parameters $\ell_0, \dots, \ell_{\log T} \in [0, \log T]$, the time complexity of Algorithm 1 when using one-level inverse maintenance is*

$$\tilde{O}\left(n^\omega + n^{5/3} \cdot \sum_{j=0}^{\log T} 2^{3\ell_j-j} + \sum_{j=0}^{\log T} \mathcal{T}_{\text{mat}}(n, n, n^{1/3} \cdot 2^{3\ell_j-j}) \cdot 2^{j-\ell_j}\right) \text{poly}(\epsilon^{-1}).$$

In particular, when $\omega = 2 + o(1)$, this time complexity is bounded by

$$\tilde{O}\left(n^{2+1/12}\right) \text{poly}(\epsilon^{-1}).$$

Proof. By Lemma C.3 there exists a matrix $\mathbf{N}(\bar{\mathbf{r}}) \in \mathbb{R}^{O(n) \times O(n)}$ such that $\Delta^{(i,k)}$ and $\mathbf{C}\Delta^{(i,k)} - \mathbf{d}$ can be read off from a column of $\mathbf{N}(\bar{\mathbf{r}}^{(i,k)})^{-1}$, and it undergoes coordinate updates to $\bar{\mathbf{r}}$. We maintain this inverse using the one-level data structure of Lemma C.1. We use the one-level inverse maintenance data structure to query for $\Delta^{(i,k)}$ and $\mathbf{u}^{(i,k)}$ exactly in each iteration, and the correctness is straightforward.

Note that the $\mathbf{r}^{(i,k)}$'s in Algorithm 1 follows the update scheme of Lemma B.4 with $\zeta = \tilde{O}(n^{1/3}\epsilon^{2/3})$. We maintain a vector $\bar{\mathbf{r}}^{(i,k)} \approx_\delta \mathbf{r}^{(i,k)}$ for Algorithm 1, and we update $\bar{\mathbf{r}}^{(i,k)}$ in a slightly different way than Line 8 of Algorithm 1: We update it by SELECTVECTORL3 (Algorithm 4) instead of SELECTVECTOR.

Let $\ell_0, \ell_1, \dots, \ell_{\log T} \in [0, \log T]$ be parameters to be determined later. For any $j \in [0 : \log T]$, we perform an update operation of the inverse maintenance data structure of Lemma C.1 in all iterations $B_j[k]$ where $k \equiv 0 \pmod{2^{\ell_j}}$. Note that by Corollary B.5 this update has size $k_j = O\left(\zeta \cdot 2^{3\ell_j-j} \cdot \frac{\log^6 n}{\delta^3}\right)$. Next we compute the time complexity of all query and update operations. and note that it follows the update scheme of Lemma B.4.

Query. The total number of updated coordinates when performing a query operation is at most

$$k = \sum_{j=0}^{\log T} k_j = \sum_{j=0}^{\log T} O\left(\zeta \cdot 2^{3\ell_j-j} \cdot \frac{\log^6 n}{\delta^3}\right)$$

Note that we can assume $k \leq n$ since otherwise the query is trivial. So the query time in each iteration is

$$O(k^\omega + nk) = n \cdot \sum_{j=0}^{\log T} \tilde{O}\left(n^{1/3} \cdot 2^{3\ell_j-j}\right) \cdot \text{poly}(\epsilon^{-1}).$$

Update. For any $j \in [0 : \log T]$, the update takes time $\mathcal{T}_{\text{mat}}(n, n, k_j)$, and in total it is performed $\frac{|B_j|}{2^{\ell_j}}$ times. So the amortized cost of all updates is

$$\frac{1}{T} \cdot \sum_{j=0}^{\log T} \mathcal{T}_{\text{mat}}(n, n, k_j) \cdot \frac{|B_j|}{2^{\ell_j}} \leq \tilde{O}\left(\frac{1}{T} \cdot \sum_{j=0}^{\log T} \mathcal{T}_{\text{mat}}(n, n, n^{1/3} \cdot 2^{3\ell_j-j}) \cdot 2^{j-\ell_j}\right) \cdot \text{poly}(\epsilon^{-1}),$$

where we used Lemma B.3 that $|B_j| \leq 2^{j+1}$.

Total runtimes. Combining the query and update operation and the $\tilde{O}(n^\omega) \text{poly}(\epsilon^{-1})$ initialization time, and since $T = \tilde{O}(n^{1/3}) \text{poly}(\epsilon^{-1})$, the total running time is

$$\begin{aligned} & \tilde{O}\left(n^\omega + n^{5/3} \cdot \sum_{j=0}^{\log T} 2^{3\ell_j-j} + \sum_{j=0}^{\log T} \mathcal{T}_{\text{mat}}(n, n, n^{1/3} \cdot 2^{3\ell_j-j}) \cdot 2^{j-\ell_j}\right) \text{poly}(\epsilon^{-1}) \\ & \leq \tilde{O}\left(n^\omega + n^{5/3} \cdot \sum_{j=0}^{\log T} 2^{3\ell_j-j} + \sum_{j=0}^{\log T} \left(n^{\frac{\omega-2}{3(1-\alpha)}} \cdot 2^{(3\ell_j-j)\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}} + n^2\right) \cdot 2^{j-\ell_j}\right) \text{poly}(\epsilon^{-1}) \\ & = \tilde{O}\left(n^\omega + n^{5/3} \cdot \sum_{j=0}^{\log T} 2^{3\ell_j-j} + \sum_{j=0}^{\log T} \left(n^{\frac{\omega-3\alpha\omega+4}{3(1-\alpha)}} \cdot 2^{\frac{3\omega+\alpha-7}{1-\alpha}\ell_j + \frac{3-\omega-\alpha}{1-\alpha}j} + n^2 \cdot 2^{j-\ell_j}\right)\right) \text{poly}(\epsilon^{-1}), \end{aligned}$$

where the second step follows from Fact A.7.

We choose ℓ_j such that $n^2 \cdot 2^{j-\ell_j} = \max\{n^{5/3} \cdot 2^{3\ell_j-j}, n^{\frac{\omega-3\alpha\omega+4}{3(1-\alpha)}} \cdot 2^{\frac{3\omega+\alpha-7}{1-\alpha}\ell_j + \frac{3-\omega-\alpha}{1-\alpha}j}\}$, i.e.,

$$2^{\ell_j} = n^{1/12} 2^{j/2} \quad \text{or} \quad n^{\alpha/3-1/9} 2^{j/3}.$$

So we have the total runtime is the minimum of the following two equations:

$$\begin{aligned} & \tilde{O}\left(n^\omega + n^{5/3} \cdot \sum_{j=0}^{\log T} n^{1/4} 2^{j/2} + \sum_{j=0}^{\log T} \left(n^{\frac{\omega-3\alpha\omega+4}{3(1-\alpha)}} \cdot n^{\frac{3\omega+\alpha-7}{12(1-\alpha)}} 2^{\frac{3\omega+\alpha-7}{2(1-\alpha)}j} \cdot 2^{\frac{3-\omega-\alpha}{1-\alpha}j}\right)\right) \text{poly}(\epsilon^{-1}) \\ & = \tilde{O}\left(n^\omega + n^{5/3} \cdot \sum_{j=0}^{\log T} n^{1/4} 2^{j/2} + \sum_{j=0}^{\log T} \left(n^{\frac{7\omega-12\alpha\omega+9+\alpha}{12(1-\alpha)}} \cdot 2^{\frac{\omega-\alpha-1}{2(1-\alpha)}j}\right)\right) \text{poly}(\epsilon^{-1}) \\ & = \tilde{O}\left(n^\omega + n^{25/12} + n^{\frac{9\omega-12\alpha\omega+7-\alpha}{12(1-\alpha)}}\right) \text{poly}(\epsilon^{-1}), \end{aligned}$$

and

$$\begin{aligned} & \tilde{O}\left(n^\omega + n^{5/3} \cdot \sum_{j=0}^{\log T} n^{\alpha-1/3} + \sum_{j=0}^{\log T} \left(n^2 \cdot 2^{2j/3} \cdot n^{-\alpha/3+1/9}\right)\right) \text{poly}(\epsilon^{-1}) \\ & = \tilde{O}\left(n^\omega + n^{4/3+\alpha} + n^{2+1/3-\alpha/3}\right) \text{poly}(\epsilon^{-1}) \\ & = \tilde{O}\left(n^\omega + n^{2+1/3-\alpha/3}\right) \text{poly}(\epsilon^{-1}). \end{aligned}$$

In conclusion, we have that the total runtime is

$$\tilde{O}\left(n^\omega + \min\left\{n^{2+1/12} + n^{\omega - \frac{3\omega + \alpha - 7}{12(1-\alpha)}}, n^{2+1/3-\alpha/3}\right\}\right) \text{poly}(\epsilon^{-1}).$$

Note that when $\omega = 2$, we have $\alpha = 1$, and the algorithm runs in $\tilde{O}(n^\omega + n^{2+1/12}) \cdot \text{poly}(\epsilon^{-1})$ time. □

F Guarantees of Algorithm 1: MWU with Monotone Weights

The algorithm in this section has been analysed previously as mentioned in the main text. We include a proof similar to that of [Adi+24] for ℓ_∞ -regression here for completeness. The analysis of Algorithm 1 is based on tracking two potential functions that were defined in Eq. (3) and (4):

$$\begin{aligned}\Phi(\mathbf{w}^{(i,k)}) &\stackrel{\text{def}}{=} \|\mathbf{w}^{(i,k)}\|_1 \\ \Psi(\bar{\mathbf{r}}^{(i,k)}) &\stackrel{\text{def}}{=} \min_{\Delta \in \mathbb{R}^d} \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\mathbf{C}\Delta - \mathbf{d})_e^2.\end{aligned}$$

Also recall that by Lemma A.1 we always have $\Psi(\bar{\mathbf{r}}^{(i,k)}) \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}^{(i,k)})$. We will show how these potential functions change with a primal step (Line 10) and a width reduction step (Line 14) in Algorithm 1. Finally, to prove our runtime bound, we will first show that if the total number of width reduction steps K is not too large, then Φ is bounded. We then prove that the number of width reduction steps cannot be too large by using the relation between Φ and Ψ and their respective changes throughout the algorithm.

Convergence Analysis

Change in Φ

Lemma F.1. *After i primal steps, and k width-reduction steps, the potential Φ is bounded as follows:*

$$\Phi(\mathbf{w}^{(i,k)}) \leq \Phi(\mathbf{w}^{(0,0)}) \left(1 + e^{\epsilon+\delta} \cdot \epsilon\alpha\right)^i \left(1 + e^{\epsilon+\delta} \cdot \frac{\epsilon}{\tau}\right)^k.$$

Proof. We prove this claim by induction. Initially, $i = k = 0$, and $\Phi(\mathbf{w}^{(0,0)}) = n$, and thus, the claim holds trivially. Assume that the claim holds for some $i, k \geq 0$. Denote $\mathbf{w} = \mathbf{w}^{(i,k)}$ and $\Delta = \Delta^{(i,k)}$.

Primal Step. If the next step is a *primal* step,

$$\Phi(\mathbf{w}^{(i+1,k)}) = \|\mathbf{w} + \epsilon\alpha|\mathbf{C}\Delta - \mathbf{d}|\mathbf{w}\|_1 = \|\mathbf{w}\|_1 + \epsilon\alpha \sum_e \mathbf{w}_e |\mathbf{C}\Delta - \mathbf{d}|_e.$$

We next bound $\sum_e |\mathbf{C}\Delta - \mathbf{d}|_e \mathbf{w}_e$. Using Cauchy-Schwarz inequality,

$$\sum_e \mathbf{w}_e |\mathbf{C}\Delta - \mathbf{d}|_e \leq \sqrt{\sum_e \mathbf{w}_e \sum_e \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2} \leq \sqrt{e^\delta \cdot \Phi(\mathbf{w}) \Psi(\bar{\mathbf{r}})} \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}), \quad (6)$$

where the last inequality follows from Lemma A.1. We thus have,

$$\Phi(\mathbf{w}^{(i+1,k)}) \leq \Phi(\mathbf{w}^{(i,k)}) \cdot \left(1 + e^{\epsilon+\delta} \cdot \epsilon\alpha\right).$$

Width Reduction Step. Let $\Delta = \Delta^{(i,k)}$ denote the solution returned in Line 9 of Algorithm 1, and let H denote the set of indices $j \in [n]$ such that $|\mathbf{C}\Delta - \mathbf{d}|_j \geq \tau$, i.e., the set of indices on which the algorithm performs width reduction. We have the following:

$$\begin{aligned} \Phi(\mathbf{w}^{(i,k+1)}) &= \sum_{j \notin H} \mathbf{w}_j^{(i,k)} + \sum_{j \in H} (\mathbf{w}_j^{(i,k)} + \epsilon \mathbf{r}_j^{(i,k)}) = \Phi + \epsilon \sum_{j \in H} \mathbf{r}_j^{(i,k)} \\ &\leq \Phi + \frac{\epsilon}{\tau} \sum_j \mathbf{r}_j^{(i,k)} |\mathbf{C}\Delta - \mathbf{d}|_j \\ &\leq \Phi(\mathbf{w}^{(i,k)}) \left(1 + \frac{\epsilon}{\tau} \cdot e^{\epsilon+\delta}\right). \end{aligned}$$

The last inequality follows from a reasoning similar to Equation (6). \square

Change in Ψ

We now prove how the Ψ potential changes throughout the algorithm.

Lemma F.2. *After i primal steps and k width reduction steps, if $\delta \leq \epsilon/6$,*

$$\Psi(\bar{\mathbf{r}}^{(i,k)}) \geq \Psi(\mathbf{r}^{(0,0)}) \left(1 + \frac{\epsilon^2 \tau^2}{4n}\right)^k.$$

Proof. First note that the primal steps only increases \mathbf{w} , and so it only increases $\Psi(\bar{\mathbf{r}})$, so it suffices to only consider the width reduction steps.

In this proof from simplicity we use \mathbf{r} and $\bar{\mathbf{r}}$ to denote $\mathbf{r}^{(i,k)}$ and $\bar{\mathbf{r}}^{(i,k)}$, and \mathbf{r}' and $\bar{\mathbf{r}}'$ to denote $\mathbf{r}^{(i,k+1)}$ and $\bar{\mathbf{r}}^{(i,k+1)}$. We let H denote the set of indices $j \in [n]$ such that $|\mathbf{C}\Delta - \mathbf{d}|_j \geq \tau$, i.e., the set of indices on which the algorithm performs width reduction.

We start by noting the following from Lemma A.3 that for $\bar{\mathbf{r}}, \bar{\mathbf{r}}' \geq 0$

$$\Psi(\bar{\mathbf{r}}') \geq \Psi(\bar{\mathbf{r}}) + \sum_e \left(1 - \frac{\bar{\mathbf{r}}_e}{\bar{\mathbf{r}}'_e}\right) \bar{\mathbf{r}}_e (\mathbf{C}\tilde{\Delta} - \mathbf{d})_e^2,$$

where $\tilde{\Delta}$ is the solution of $\Psi(\bar{\mathbf{r}})$. For any coordinate $e \in H$,

$$\frac{\mathbf{r}_e^{(i,k+1)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} \geq \frac{\mathbf{w}_e^{(i,k+1)} - \mathbf{w}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} = \epsilon \frac{\mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} \geq \frac{\epsilon}{1 + 2\epsilon}.$$

In the above we used, $\mathbf{w}_e^{(i,k+1)} = \mathbf{w}_e^{(i,k)} + \epsilon \mathbf{r}_e^{(i,k)}$, and $\mathbf{r}_e^{(i,k+1)} \leq (1 + 2\epsilon) \mathbf{r}_e^{(i,k)}$. Now,

$$\frac{\bar{\mathbf{r}}'_e - \bar{\mathbf{r}}_e}{\bar{\mathbf{r}}'_e} \geq \frac{e^{-\delta} \mathbf{r}'_e - e^{\delta} \mathbf{r}_e}{e^{\delta} \mathbf{r}'_e} = \frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}'_e} - \frac{e^{\delta} - e^{-\delta}}{e^{\delta}} \geq \frac{\epsilon}{1 + 2\epsilon} - 2\delta.$$

Since $\delta \leq \epsilon/6$, the above becomes,

$$\frac{\bar{\mathbf{r}}'_e - \bar{\mathbf{r}}_e}{\bar{\mathbf{r}}'_e} \geq \frac{\epsilon}{3}.$$

We also know that for all $e \in H$, $|\mathbf{C}\Delta - \mathbf{d}|_e \geq \tau$, and since $\bar{\mathbf{r}}_e \geq e^{-\delta} \mathbf{r}_e \geq e^{-\delta} \frac{\epsilon}{n} \Phi(\mathbf{w})$,

$$\Psi(\bar{\mathbf{r}}') \geq \Psi(\bar{\mathbf{r}}) + \frac{\epsilon}{3} \cdot e^{-\delta} \frac{\epsilon}{n} \Phi(\mathbf{w}) \cdot \tau^2 \geq \Psi(\bar{\mathbf{r}}) + \frac{\epsilon^2 \tau^2 e^{-\epsilon-2\delta}}{3n} \Psi(\bar{\mathbf{r}}).$$

Overall, after k such steps and using that $\delta \leq \epsilon/6 \leq 1/60$, we get,

$$\Psi(\bar{\mathbf{r}}^{(i,k)}) \geq \Psi(\mathbf{r}^{(0,0)}) \left(1 + \frac{\epsilon^2 \tau^2}{4n}\right)^k.$$

\square

Proof of Theorem 2.1

Proof. Let $\hat{\mathbf{x}} = \frac{\mathbf{x}^{(T)}}{T}$ be the solution returned by Algorithm 1. We first bound the objective value at $\hat{\mathbf{x}}$. Suppose the algorithm terminates in $T = \alpha^{-1}\epsilon^{-2}\log n$ primal steps and $K \leq \tau/\epsilon^2$ width reduction steps. We can assume this without loss of generality since otherwise we can just halt the algorithm after executing τ/ϵ^2 width reduction steps. In the end we will show that the algorithm executes $< \tau/\epsilon^2$ width reduction steps.

We can now apply Lemma F.1 to get,

$$\Phi(\mathbf{w}^{(T,K)}) \leq ne^{2(1+\epsilon)\epsilon\alpha T + 2(1+\epsilon)\epsilon\tau^{-1}K} \leq ne^{2\frac{(1+\epsilon)\log n}{\epsilon} + (1+\epsilon)} \leq ne^{\frac{2(1+\epsilon)}{\epsilon}(1+\log n)} = n^{O(\frac{1}{\epsilon})}.$$

We next observe from the weight and \mathbf{x} update steps in our algorithm that, $\mathbf{w}^{(T,K)} \geq |\mathbf{x}^{(T)}|/T$.

$$\mathbf{w}_e^{(T,K)} \geq \prod_{i \geq 0} \left(1 + \epsilon\alpha |\mathbf{C}\Delta^{(i,k)} - \mathbf{d}|_e\right) \geq \exp\left((1-\epsilon)\epsilon\alpha \sum_{i \geq 0} |\mathbf{C}\Delta^{(i,k)} - \mathbf{d}|_e\right).$$

In the last inequality we used that $\|\mathbf{C}\Delta^{(i,k)} - \mathbf{d}\|_\infty \leq \tau$ and $\alpha\tau \leq 1$ along with $\exp(\epsilon(1-\epsilon)x) \leq (1+\epsilon x)$ for $0 < \epsilon < 1/2$ and $0 \leq x \leq 1$. We also know that $\mathbf{w}_e^{(T,K)} \leq \Phi(\mathbf{w}^{(T,K)}) \leq ne^{(1+\delta)(1+\epsilon)\epsilon\alpha T + (1+\delta)(1+\epsilon)/\epsilon}$. Now,

$$\|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty \leq \frac{1}{T} \sum_{t=1}^T \|\mathbf{C}\Delta^{(t,k)} - \mathbf{d}\|_\infty \leq \frac{(1+\epsilon)(1+\delta)}{1-\epsilon} + \frac{(1+\epsilon)(1+\delta)}{\epsilon\alpha T(1-\epsilon)} + \frac{\log n}{(1-\epsilon)\epsilon\alpha T} \leq 1 + 10\epsilon.$$

We have shown that if the number of width reduction steps is bounded by K then our algorithm returns the required solution. We will next prove that we cannot have more than K width reduction steps.

Suppose to the contrary, the algorithm takes a width reduction step starting from step (i, k) where $i < T$ and $k = \tau/\epsilon^2$. Since the conditions for Lemma F.1 hold for all preceding steps, we must have $\Phi(\mathbf{w}^{(i,k)}) \leq n^{O(\frac{1}{\epsilon})}$ which combined with Lemma A.1 implies $\Psi \leq (1+\epsilon)n^{O(\frac{1}{\epsilon})}$. Let $L = \mathbf{d}^\top (\mathbf{I} - \mathbf{C}^\top (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}) \mathbf{d}$ denote a lower bound on $\Psi(\mathbf{r}^{(0,0)})$. Using this bound, from lemma F.2,

$$\Psi(\bar{\mathbf{r}}^{(i,k+1)}) \geq \Psi(\mathbf{r}^{(0,0)}) \left(1 + \frac{\epsilon^2 \tau^2}{4(1+\epsilon)(1+\delta)n}\right)^k \geq L \left(1 + \frac{\epsilon^2 \tau^2}{4(1+\epsilon)(1+\delta)n}\right)^k.$$

Therefore,

$$n^{O(\frac{1}{\epsilon})} \geq L \left(1 + \frac{\epsilon^2 \tau^2}{(1+\epsilon)(1+\delta)n}\right)^k.$$

which is a contradiction if $K > \tau/\epsilon^2$ for the set value of $\tau = \Theta\left(\frac{n^{\frac{1}{3}}}{\epsilon^{\frac{1}{3}}} \log \frac{n}{L}\right)$. We can thus conclude that we can never have more than $K = \tau/\epsilon^2$ width reduction steps, thus concluding the correctness of the returned solution. The total number of iterations is at most,

$$T + K \leq \alpha^{-1}\epsilon^{-2}\log n + \tau\epsilon^{-2} = \Theta\left(n^{1/3}\epsilon^{-7/3}\log^2 \frac{n}{L}\right).$$

□

G Guarantees of Algorithm 3: Robust Primal Step and Stable Width Reduction Step

G.1 Starting Point: Non-Monotone MWU Algorithm

We first present the starting point from where we developed our final algorithm. We extend the algorithm in [Mad16] to general ℓ_∞ -regression. This gives an algorithm which converges in $\tilde{O}(n^{1/3} \text{poly}(\epsilon^{-1}))$ iterations and has weights that are not monotonically increasing. The algorithm is presented as Algorithm 10 and we have moved the analysis to the end in Appendix J since we do not really use this algorithm directly. The analysis is just kept for completeness. In particular we prove:

Theorem G.1. *Let $\eta \leq 1/6$. There is an algorithm that does not update the weights monotonically (Algorithm 10, and with input $\begin{bmatrix} \mathbf{C} \\ -\mathbf{C} \end{bmatrix}, \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix}, \epsilon$) returns $\hat{\mathbf{x}}$ such that $\|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty \leq 1 + O(\epsilon)$ in at most $\tilde{O}\left((n^{1/2-\eta} + n^{2\eta})\epsilon^{-7/3}\right)$ iterations. Each iteration solves a system of linear equations.*

Algorithm 10 Accelerated MWU algorithm with non-monotone weights

```

1: procedure MWU-NONMONOTONE( $\tilde{\mathbf{C}}, \tilde{\mathbf{d}}, \epsilon$ )
2:    $\mathbf{w}^{(0,0)} \leftarrow \mathbf{1}_n, \quad \mathbf{x}^{(0,0)} \leftarrow \mathbf{0}_d$ 
3:    $\alpha \leftarrow \tilde{\Theta}\left(n^{-1/2+\eta}\epsilon^{1/3}\right), \alpha_+ \leftarrow \alpha, \alpha_- \leftarrow \alpha/(1+2\epsilon)$ 
4:    $\tau \leftarrow \tilde{\Theta}\left(n^{1-4\eta}\epsilon^{-1/3}\right), \quad \rho \leftarrow \tilde{\Theta}\left(n^{1/2-3\eta}\right)$ 
5:    $T \leftarrow \alpha^{-1} \ln \frac{n}{\Psi_0}/\epsilon^2$ 
6:    $i, k = 0$ 
7:   while  $i < T$  do
8:      $\mathbf{r}^{(i,k)} \leftarrow \mathbf{w}^{(i,k)} + \frac{\epsilon}{2n} \sum_e \mathbf{w}_e^{(i,k)}$ 
9:      $\Delta^{(i,k)} \leftarrow \arg \min_{\Delta} \sum_e \mathbf{r}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e^2 \quad \triangleright \Delta = (\tilde{\mathbf{C}}^\top \mathbf{R}^{(i,k)} \tilde{\mathbf{C}})^{-1} \tilde{\mathbf{C}}^\top \mathbf{R}^{(i,k)} \tilde{\mathbf{d}}$ 
10:     $\Psi(\mathbf{r}^{(i,k)}) \leftarrow \sum_e \mathbf{r}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2$ 
11:    if  $\sum_e \mathbf{r}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \leq 2\rho\Psi(\mathbf{r}^{(i,k)})$  then  $\triangleright$  primal step
12:       $\vec{\alpha}_e^{(i,k)} = \begin{cases} \alpha_+ & \text{if } (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e \geq 0 \\ \alpha_- & \text{else} \end{cases}$ 
13:       $\mathbf{w}^{(i+1,k)} \leftarrow \mathbf{w}^{(i,k)} \left(1 + \epsilon \vec{\alpha}^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})\right)$ 
14:       $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \Delta^{(i,k)}$ 
15:       $i \leftarrow i + 1$ 
16:    else  $\triangleright$  width reduction step
17:      Let  $S$  be the set of coordinates  $e$  such that  $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \geq \rho$ 
18:       $H \subseteq S$  be maximal subset such that  $\sum_{e \in H} \mathbf{r}_e^{(i,k)} \leq \tau^{-1}\Psi(\mathbf{r}^{(i,k)})$ 
19:      For all  $e \in H, \mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon)\mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{n}\Phi(\mathbf{w}^{(i,k)})$ 
20:      If  $H \neq S$ , for one  $\bar{e} \in S \setminus H$ , let  $\gamma = \min\{1, \frac{\tau^{-1}}{\mathbf{r}_{\bar{e}}^{(i,k)}}\Psi(\mathbf{r}^{(i,k)})\}$ 
21:       $\mathbf{w}_{\bar{e}}^{(i,k+1)} \leftarrow (1 + \epsilon\gamma)\mathbf{w}_{\bar{e}}^{(i,k)} + \frac{\epsilon^2\gamma}{n}\Phi(\mathbf{w}^{(i,k)})$ 
22:       $k \leftarrow k + 1$ 
23:  return  $\mathbf{x}^{(T)}/T$ 

```

Let us consider the width reduction steps. Since $\mathbf{r}_e \geq \frac{\epsilon}{2n}\Phi(\mathbf{w}) \geq \frac{\epsilon}{2n}\Psi(\mathbf{w})$, from the condition

of the width steps we get that,

$$\frac{\epsilon}{2n} \Psi(\mathbf{r}) |H| \leq \tau^{-1} \Psi(\mathbf{r}) \Rightarrow |H| \leq O(n^{4\eta}).$$

The above calculations imply that for $\eta = 1/6$, the algorithm in the worst case can update $\approx n^{2/3}$ coordinates in every iteration. There are a total of $\approx n^{1/3}$ iterations. At an intuitive level, if we change the value of \mathbf{r} so many coordinates per iteration, and that too by a factor of ϵ , then the entire idea of doing a lazy update is moot. Therefore, we require a new kind of width reduction steps that can schedule these steps in a way amenable to a lazy update schedule, which is what we do in the next section.

G.2 Warm Up: Stable Width Reduction Step

As a warm-up, we first analyze the required stable algorithm, but do not introduce any sketching. In the next section, we will consider the changes to the analysis from this section which occur due to the introduction of sketching. We encourage the reader to go through this section in order to understand how our final algorithm works.

In this section we define $\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} \\ -\mathbf{C} \end{bmatrix}$ and $\tilde{\mathbf{d}} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix}$. Our goal is to prove the following result. We will focus on the convergence and runtime in this section and defer the stability guarantees of the algorithm to Appendix H.

Theorem G.2. *For $\eta \leq 1/10$, Algorithm 11 with input $(\begin{bmatrix} \mathbf{C} \\ -\mathbf{C} \end{bmatrix}, \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix}, \epsilon)$ finds $\hat{\mathbf{x}} \in \mathbb{R}^d$ such that $\|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty \leq 1 + O(\epsilon)$ in at most $T + K \leq \tilde{O}(n^{1/2-\eta}\epsilon^{-4})$ iterations. Furthermore, the algorithm satisfies the following additional guarantees:*

1. *In the width reduction step of the algorithm, the algorithm only requires to find at most $\tilde{O}(n^{1/2+\eta})$ large coordinates per iteration.*
2. *The algorithm satisfies the following low-rank update scheme: There are at most $\frac{T+K}{2^\ell}$ number of iterations where $\bar{\mathbf{r}}$ receives an update of rank $\tilde{O}_\epsilon(n^{1/5}2^{2\ell})$.*

We first define some notation and basic properties.

G.2.1 Definitions and Basic Properties

Potentials. Recall that we had defined the two potentials of interest as,

$$\Phi(\mathbf{w}) = \|\mathbf{w}\|_1, \quad \Psi(\mathbf{r}) = \min_{\Delta \in \mathbb{R}^d} \sum_{e=1}^{2n} \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2.$$

Also recall that by Lemma A.1, for $\mathbf{r} = \mathbf{w} + \frac{\epsilon}{2n} \|\mathbf{w}\|_1$, and $\bar{\mathbf{r}} \approx_\delta \mathbf{r}$, we always have $\Psi(\bar{\mathbf{r}}) \approx_\delta \Psi(\mathbf{r})$, and $\Psi(\bar{\mathbf{r}}) \leq e^{\epsilon+\delta} \Phi(\mathbf{w})$.

Lazy updates for primal steps. Recall that in Algorithm 2, for any primal step i and any $e \in [2n]$, we defined $\text{LASTWIDTH}(i, e)$ to be the largest $i' \leq i$ such that the algorithm executed a width reduction step from (i', k) to $(i', k+1)$ during which the weight of e is updated, i.e., $\mathbf{w}_e^{(i', k+1)} \neq \mathbf{w}_e^{(i', k)}$.

Algorithm 11 Accelerated MWU algorithm with non-monotone weights and stable steps

```

1: procedure MWU-NONMONOTONESTABLE( $\tilde{\mathbf{C}}, \tilde{\mathbf{d}}, \epsilon$ )
2:    $\mathbf{w}^{(0,0)} \leftarrow \mathbf{1}_{2n}$ ,  $\bar{\mathbf{r}}^{(0,0)} \leftarrow \mathbf{r}^{(0,0)} \leftarrow (1 + \epsilon)\mathbf{1}_{2n}$ ,  $\mathbf{x}^{(0)} \leftarrow \mathbf{0}_d$ 
3:    $\alpha \leftarrow n^{-1/2+\eta} \cdot \epsilon \cdot \log(n)^{-4/3} \log(\frac{n}{\Psi_0})^{-1/3}/10$ ,  $\alpha_+ \leftarrow \alpha$ ,  $\alpha_- \leftarrow \alpha/(1 + 2\epsilon)$ 
4:    $\tau \leftarrow n^{1/2-\eta} \cdot \epsilon^{-4} \cdot \log(n)^8 \log(\frac{n}{\Psi_0})^2$ ,  $\rho \leftarrow n^{1/2-3\eta} \cdot \epsilon^{-2} \cdot \log(n)^4 \log(\frac{n}{\Psi_0})$ ,  $\delta \leftarrow \frac{\epsilon}{100}$ 
5:    $T \leftarrow \alpha^{-1} \epsilon^{-2} \log n$ 
6:    $i, k = 0$ 
7:   while  $i < T$  do
8:      $\Delta^{(i,k)} \leftarrow \arg \min_{\Delta} \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e^2$   $\triangleright \bar{\mathbf{r}} \approx_{\delta} \mathbf{r}$ 
9:      $\Psi(\bar{\mathbf{r}}^{(i,k)}) \leftarrow \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2$ 
10:    if  $\sum_e \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \leq 2\rho\Psi(\bar{\mathbf{r}}^{(i,k)})$  then  $\triangleright$  primal step
11:       $\vec{\alpha}_e^{(i,k)} = \begin{cases} \alpha_+ & \text{if } (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e \geq 0 \\ \alpha_- & \text{else} \end{cases}$ 
12:       $\mathbf{w}^{(i+1,k)} \leftarrow \mathbf{w}^{(i,k)} \left(1 + \epsilon \vec{\alpha}^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})\right)$ 
13:       $\mathbf{r}^{(i+1,k)} \leftarrow \mathbf{w}^{(i+1,k)} + \frac{\epsilon}{2n} \sum_e \mathbf{w}_e^{(i+1,k)}$ 
14:       $\bar{\mathbf{r}}^{(i+1,k)} \leftarrow \text{SELECTVECTOR}(\mathbf{r}^{(i+1,k)}, i + 1, \delta)$   $\triangleright$  Algorithm 2
15:       $\mathbf{x}^{(i+1)} \leftarrow \mathbf{x}^{(i)} + \Delta^{(i,k)}$ 
16:       $i \leftarrow i + 1$ 
17:    else  $\triangleright$  width reduction step
18:      Let  $S$  be the set of coordinates  $e$  such that  $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \geq \rho$ 
19:       $H \subseteq S$  be maximal subset such that  $\sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1}\Psi(\bar{\mathbf{r}}^{(i,k)})$ 
20:      if  $H \neq S$  then
21:        Pick any  $\bar{e} \in S \setminus H$ .
22:        For all  $e \in H \cup \{\bar{e}\}$ ,  $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon)\mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n}\Phi(\mathbf{w}^{(i,k)})$ 
23:         $\mathbf{r}^{(i,k+1)} \leftarrow \mathbf{w}^{(i,k+1)} + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k+1)})$ 
24:        For all  $e \in H \cup \{\bar{e}\}$ ,  $\bar{\mathbf{r}}_e^{(i,k+1)} \leftarrow \mathbf{r}_e^{(i,k+1)}$ 
25:      else
26:        for  $\zeta = \rho, 2\rho, 4\rho, \dots, 2^{c_\rho}\rho$  do
27:           $\triangleright c_\rho$  is defined to be the smallest integer  $c$  that satisfies  $2^c\rho \geq \sqrt{n/\epsilon}$ 
28:          Define the set  $H_\zeta = \{e \in H \mid |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \in [\zeta, 2\zeta)\}$ .
29:          If  $\sum_{e \in H_\zeta} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq \frac{\rho\Psi(\bar{\mathbf{r}}^{(i,k)})}{\log(\frac{n}{\epsilon\rho})}$ , set  $\zeta^* \leftarrow \zeta$ , and break.
30:          For all  $e \in H_{\zeta^*}$ ,  $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon)\mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n}\Phi(\mathbf{w}^{(i,k)})$ 
31:           $\mathbf{r}^{(i,k+1)} \leftarrow \mathbf{w}^{(i,k+1)} + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k+1)})$ 
32:          For all  $e \in H_{\zeta^*}$ ,  $\bar{\mathbf{r}}_e^{(i,k+1)} \leftarrow \mathbf{r}_e^{(i,k+1)}$ 
33:         $k \leftarrow k + 1$ 
34:    return  $\mathbf{x}^{(T)}/T$ 

```

For any primal step i and any $e \in [2n]$, we also define $\text{LAST}(i, e)$ to be the largest $i'' \leq i$ such that $\bar{\mathbf{r}}_e$ was updated by **SELECTVECTOR** in primal step $i'' - 1$, i.e., $\bar{\mathbf{r}}_e^{(i'', k_{i''-1})} \neq \bar{\mathbf{r}}_e^{(i''-1, k_{i''-1})}$.

For any primal step i , we define $S_i \subseteq [2n]$ to be the set of coordinates that are being updated by **SELECTVECTOR** in the i -th primal step.

Finally, for any primal step i and any $e \in S_i$, we also define $\ell_{i,e}$ to be the smallest integer ℓ such that $i \equiv 0 \pmod{2^\ell}$ and $\left| \ln \left(\frac{\mathbf{r}_e^{(i)}}{\mathbf{r}_e^{(i-2^\ell)}} \right) \right| \geq \frac{\delta}{2 \log n}$.

Remark: We note that $\mathbf{r} = \mathbf{w} + \frac{\epsilon}{2n} \Phi(\mathbf{w})$. Now in our lazy update scheme, \mathbf{r} can also change due to changes in $\Phi(\mathbf{w})$. We claim that we do not need to consider the changes in \mathbf{r} due to the change in Φ in the lazy update scheme. This is because, the change in Φ contributes enough only when the change is $\approx n$ which can happen only $\tilde{O}_\epsilon(1)$ times (Refer to Lemma G.3), or once every $\tilde{O}_\epsilon(\alpha^{-1})$ iterations, and in such cases, the algorithm can reset the values of \mathbf{r} for all coordinates.

Sizes of width reduction steps. We say a width reduction step has size s if it updates the weight of s coordinates, and we denote the size of the k -th width reduction step as $\text{SIZE}(k)$. For example, if in the k -th width reduction step on Line 20 of Algorithm 11 we have that $H \neq S$, then we have $\text{SIZE}(k) = |H| + 1$ by Line 22 of Algorithm 11.

G.2.2 Change in Φ

Lemma G.3 (Change in Φ for Algorithm 11). *After i primal steps, and k width-reduction steps, if $\alpha \rho^{1/3} \leq \frac{\epsilon^{1/3}}{10n^{1/3}}$, the potential Φ is bounded as follows:*

$$\Phi(\mathbf{w}^{(i,k)}) \leq \Phi(\mathbf{w}^{(0,0)}) \cdot \left(1 + \epsilon \alpha_+ e^{\epsilon+\delta}\right)^i \cdot \left(1 + \epsilon e^{\epsilon+2\delta} \cdot (\tau^{-1} + \rho^{-2})\right)^k.$$

Furthermore, after every primal step, the potential can decrease by at most,

$$\Phi(\mathbf{w}^{(i+1,k)}) \geq \Phi(\mathbf{w}^{(i,k)}) \left(1 - \epsilon \alpha_+ e^{\epsilon+\delta}\right).$$

Proof. First note that we always have $\mathbf{w}_e^{(i,k)} > 0$ from the same proof as Lemma J.1.

Primal Step. If the next step is a *primal* step, then

$$\begin{aligned} \Phi(\mathbf{w}^{(i+1,k)}) &= \left\| \mathbf{w}^{(i,k)} + \epsilon \vec{\alpha}^{(i,k)} (\tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}}) \mathbf{w}^{(i,k)} \right\|_1 \\ &= \|\mathbf{w}^{(i,k)}\|_1 + \epsilon \sum_e \mathbf{w}_e^{(i,k)} \vec{\alpha}_e^{(i,k)} (\tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}})_e. \end{aligned} \quad (7)$$

We first bound $\sum_e \mathbf{w}_e^{(i,k)} \cdot \vec{\alpha}_e^{(i,k)} \cdot (\tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}})_e$. Using Cauchy-Schwarz inequality, we have

$$\begin{aligned} \sum_e \mathbf{w}_e^{(i,k)} \cdot \vec{\alpha}_e^{(i,k)} \cdot |\tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}}|_e &\leq \sqrt{\left(\sum_e \mathbf{w}_e^{(i,k)} (\vec{\alpha}_e^{(i,k)})^2 \right) \cdot \left(\sum_e \mathbf{w}_e^{(i,k)} \cdot (\tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \right)} \\ &\leq \alpha_+ \cdot \sqrt{e^\delta \cdot \Phi(\mathbf{w}^{(i,k)}) \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})} \\ &\leq \alpha_+ \cdot e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}^{(i,k)}), \end{aligned}$$

where the second step follows from $\alpha_- < \alpha_+$ and $\mathbf{w}_e^{(i,k)} \leq e^\delta \cdot \bar{\mathbf{r}}_e^{(i,k)}$, the third step follows from Lemma A.1 that $\Psi(\bar{\mathbf{r}}^{(i,k)}) \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}^{(i,k)})$.

Now, from Equation (7), and $\mathbf{w}_e^{(i,k)} > 0$, we have that $\Phi(\mathbf{w}^{(i+1,k)})$ is between $\Phi(\mathbf{w}^{(i,k)}) - \epsilon \sum_e \mathbf{w}_e^{(i,k)} \vec{\alpha}_e^{(i,k)} | \tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}} |_e$ and $\Phi(\mathbf{w}^{(i,k)}) + \epsilon \sum_e \mathbf{w}_e^{(i,k)} \vec{\alpha}_e^{(i,k)} | \tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}} |_e$.

Therefore, we get our bounds,

$$\Phi(\mathbf{w}^{(i,k)})(1 - \epsilon \alpha_+ e^{\epsilon+\delta}) \leq \Phi(\mathbf{w}^{(i+1,k)}) \leq \Phi(\mathbf{w}^{(i,k)})(1 + \epsilon \alpha_+ e^{\epsilon+\delta}).$$

Width Reduction Step.

When $H \neq S$, we have the following:

$$\begin{aligned} \Phi(\mathbf{w}^{(i,k+1)}) &= \sum_{j \notin H \cup \{\bar{e}\}} \mathbf{w}_j^{(i,k)} + \sum_{j \in H \cup \{\bar{e}\}} \left((1 + \epsilon) \mathbf{w}_j^{(i,k)} + \frac{\epsilon^2}{2n} \Phi(\mathbf{w}^{(i,k)}) \right) \\ &= \Phi(\mathbf{w}^{(i,k)}) + \epsilon \sum_{j \in H \cup \{\bar{e}\}} \mathbf{r}_j^{(i,k)} \\ &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon e^\delta \sum_{j \in H} \bar{\mathbf{r}}_j^{(i,k)} + \epsilon e^\delta \bar{\mathbf{r}}_{\bar{e}}^{(i,k)} \\ &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon e^\delta \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)}) + \epsilon e^\delta \rho^{-2} \Psi(\bar{\mathbf{r}}^{(i,k)}) \\ &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon e^{\epsilon+2\delta} (\tau^{-1} + \rho^{-2}) \Phi(\mathbf{w}^{(i,k)}) \end{aligned}$$

where the fourth step follows from $\sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)})$ by the definition of H , and that $\bar{\mathbf{r}}_{\bar{e}}^{(i,k)} \leq \frac{\Psi(\bar{\mathbf{r}}^{(i,k)})}{(\tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}})_{\bar{e}}^2} \leq \frac{\Psi(\bar{\mathbf{r}}^{(i,k)})}{\rho^2}$ since $\bar{e} \in S$ and so it satisfies $|\tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}}|_{\bar{e}} \geq \rho$, and the last step follows from $\Psi(\bar{\mathbf{r}}^{(i,k)}) \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}^{(i,k)})$ by Lemma A.1.

Now, when $H = S$, we only update weights in a set $H_{\zeta^*} \subseteq H$ (see Line 30 in Algorithm 11),

$$\begin{aligned} \Phi(\mathbf{w}^{(i,k+1)}) &= \sum_{j \notin H_{\zeta^*}} \mathbf{w}_j^{(i,k)} + \sum_{j \in H_{\zeta^*}} \left((1 + \epsilon) \mathbf{w}_j^{(i,k)} + \frac{\epsilon^2}{2n} \Phi(\mathbf{w}^{(i,k)}) \right) \\ &= \Phi(\mathbf{w}^{(i,k)}) + \epsilon \sum_{j \in H_{\zeta^*}} \bar{\mathbf{r}}_j^{(i,k)} \\ &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)}) \\ &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon e^{\epsilon+\delta} \cdot \tau^{-1} \Phi(\mathbf{w}^{(i,k)}), \end{aligned}$$

where the third step follows from $\sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} \leq \sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)})$ by the definition of H , and the fourth step follows from $\Psi(\bar{\mathbf{r}}^{(i,k)}) \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}^{(i,k)})$.

In both cases we have

$$\Phi(\mathbf{w}^{(i,k+1)}) \leq \Phi(\mathbf{w}^{(i,k)}) (1 + \epsilon e^{\epsilon+2\delta} \cdot (\tau^{-1} + \rho^{-2})).$$

Also note that for a width reduction step, $\Phi(\mathbf{w}^{(i,k+1)}) \geq \Phi(\mathbf{w}^{(i,k)})$. □

G.2.3 Change in Ψ

Lemma G.4 (Bound on $|H|$ in width reduction steps). *For any width reduction step, the size of H satisfies $|H| \leq \frac{n}{\tau\epsilon} \cdot e^{\epsilon+2\delta}$.*

Proof. Consider a width reduction step which updates (i, k) to $(i, k + 1)$.

Since $\sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)})$ and $\bar{\mathbf{r}}_e^{(i,k)} \geq e^{-\delta} \mathbf{r}_e^{(i,k)} \geq e^{-\delta} \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i,k)}) \geq e^{-\epsilon-2\delta} \frac{\epsilon}{2n} \Psi(\bar{\mathbf{r}}^{(i,k)})$, we have

$$\begin{aligned} |H| \cdot e^{-\epsilon-2\delta} \cdot \frac{\epsilon}{2n} \Psi(\bar{\mathbf{r}}^{(i,k)}) &\leq \sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)}) \\ \Rightarrow |H| &\leq \frac{n}{\tau \cdot \epsilon} \cdot e^{\epsilon+2\delta}. \end{aligned} \quad \square$$

Let L be the largest power of 2 such that $L \leq \frac{1}{100(\log^4 n) \epsilon \alpha \rho}$. Note that we have $L = \Theta(\frac{1}{(\log^4 n) \epsilon \alpha \rho})$. We have the following lemma.

Lemma G.5 (Change in Ψ for Algorithm 11). *For any integer $c \geq 0$, after L primal steps from $(c-1)L$ to cL , if $\rho^2 \tau^{-1} \geq 0.1$, the potential Ψ is bounded as follows:*

$$\Psi(\bar{\mathbf{r}}^{(cL, k_{cL})}) \geq \Psi(\bar{\mathbf{r}}^{((c-1)L, k_{(c-1)L})}) \cdot \left(1 - \tilde{O}(\epsilon \alpha \rho L)\right) \cdot \prod_{k=k_{(c-1)L}}^{k_{cL}} \left(1 + O\left(\frac{\epsilon^{4/3} \rho^{2/3} \cdot \text{SIZE}(k)^{1/3}}{n^{1/3} \cdot \log^{2/3}(\frac{n}{\epsilon \rho})}\right)\right).$$

Proof. Width Reduction Step. We first consider the width reduction steps. Note that if the width step updates a coordinate e , then it always updates $\bar{\mathbf{r}}_e^{(i,k+1)}$ to its exact value $\mathbf{r}_e^{(i,k+1)}$ (see Line 24 and 32 in Algorithm 11). So we always have $\bar{\mathbf{r}}_e^{(i,k+1)} = \mathbf{r}_e^{(i,k+1)}$ if e is being updated by the k -th width reduction step, and otherwise we have $\bar{\mathbf{r}}_e^{(i,k+1)} = \bar{\mathbf{r}}_e^{(i,k)}$.

Next we prove an upper bound on $\mathbf{r}_e^{(i,k+1)}$. If the width reduction step updates the weight of a coordinate e to be $\mathbf{w}_e^{(i,k+1)} \leftarrow (1 + \epsilon) \mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n} \Phi(\mathbf{w}^{(i,k)})$, then we have

$$\begin{aligned} \mathbf{r}_e^{(i,k+1)} &= \mathbf{w}_e^{(i,k+1)} + \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i,k+1)}) \\ &= (1 + \epsilon) \mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n} \Phi(\mathbf{w}^{(i,k)}) + \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i,k+1)}) \\ &= (1 + \epsilon) \mathbf{r}_e^{(i,k)} + \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i,k+1)}) - \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i,k)}) \\ &\leq (1 + \epsilon) \mathbf{r}_e^{(i,k)} + \frac{\epsilon}{2n} 2\epsilon \cdot \Phi(\mathbf{w}^{(i,k+1)}) \\ &\leq (1 + 3\epsilon) \mathbf{r}_e^{(i,k)}, \end{aligned} \quad (8)$$

where the fourth step follows from $\Phi(\mathbf{w}^{(i,k+1)}) \leq \Phi(\mathbf{w}^{(i,k)}) \cdot (1 + \epsilon e^{\epsilon+2\delta} (\tau^{-1} + \rho^{-2})) \leq \Phi(\mathbf{w}^{(i,k)}) \cdot (1 + 2\epsilon)$ by Lemma G.3. Note that we also have $\mathbf{r}_e^{(i,k+1)} \geq \mathbf{r}_e^{(i,k)}$ since we are only increasing the weights. Also,

$$\begin{aligned} \frac{\bar{\mathbf{r}}_e^{(i,k+1)} - \bar{\mathbf{r}}_e^{(i,k)}}{\bar{\mathbf{r}}_e^{(i,k+1)}} &\geq \frac{\mathbf{r}_e^{(i,k+1)} - (1 + 0.1\epsilon) \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} \\ &\geq \frac{\mathbf{w}_e^{(i,k+1)} - \mathbf{w}_e^{(i,k)} - 0.1\epsilon \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} \\ &= \frac{(1 + \epsilon) \mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{2n} \Phi(\mathbf{w}^{(i,k)}) - \mathbf{w}_e^{(i,k)} - 0.1\epsilon \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} \\ &= \frac{0.9\epsilon \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} \geq \frac{0.9\epsilon}{1 + 3\epsilon}. \end{aligned} \quad (9)$$

where the first step follows from $\bar{\mathbf{r}}_e^{(i,k+1)} = \mathbf{r}_e^{(i,k+1)}$, and $\bar{\mathbf{r}}_e^{(i,k)} \leq e^\delta \mathbf{r}_e^{(i,k)} \leq (1 + 0.1\epsilon) \mathbf{r}_e^{(i,k)}$ since $\delta = \epsilon/100$, and the last step follows from Eq. (8).

Next we consider the two cases of $H \neq S$ and $H = S$ separately.

When $H \neq S$. Using Lemma A.3 we have the following:

$$\begin{aligned}
\Psi(\bar{\mathbf{r}}^{(i,k+1)}) &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \sum_e \left(\frac{\bar{\mathbf{r}}_e^{(i,k+1)} - \bar{\mathbf{r}}_e^{(i,k)}}{\bar{\mathbf{r}}_e^{(i,k+1)}} \right) \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \\
&\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \frac{0.9\epsilon}{1+3\epsilon} \cdot \sum_{e \in H \cup \{\bar{e}\}} \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \\
&\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \frac{0.9\epsilon}{1+3\epsilon} \cdot \rho^2 \cdot \sum_{e \in H \cup \{\bar{e}\}} \bar{\mathbf{r}}_e^{(i,k)} \\
&\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \frac{0.9\epsilon}{1+3\epsilon} \cdot \rho^2 \cdot \tau^{-1} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)}) \\
&= \Psi(\bar{\mathbf{r}}^{(i,k)}) \cdot \left(1 + O(\epsilon \rho^2 \tau^{-1}) \right) \geq \Psi(\bar{\mathbf{r}}^{(i,k)}) \cdot \left(1 + O\left(\frac{\epsilon^{4/3} \rho^{2/3} \cdot |H|^{1/3}}{n^{1/3}} \right) \right),
\end{aligned}$$

where the second step follows from when $H \neq S$ the width reduction step only updates edges in $H \cup \{\bar{e}\}$ and Eq. (9), the third step follows from every $e \in S$ satisfies $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \geq \rho$ and $H \cup \{\bar{e}\} \subseteq S$, the fourth step follows from $H \subseteq S$ is a maximal subset such that $\sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} \leq \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)})$ so we must have $\sum_{e \in H \cup \{\bar{e}\}} \bar{\mathbf{r}}_e^{(i,k)} \geq \tau^{-1} \Psi(\bar{\mathbf{r}}^{(i,k)})$, and the last step follows from Lemma G.4 that $|H| \leq \frac{n}{\tau\epsilon} \cdot e^{\epsilon+2\delta}$ and our assumption $\rho^2 \tau^{-1} \geq 0.1$ and so $\rho^2 \tau^{-1} \geq 0.2 \cdot \rho^{2/3} \tau^{-1/3}$.

When $H = S$. In this case any $e \notin H = S$ must satisfy $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e < \rho$, so we have

$$\sum_{e \notin H} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \leq \max_{e \notin H} \{|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e\} \sum_{e \notin H} \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \leq \rho \Psi(\bar{\mathbf{r}}^{(i,k)}).$$

Since this is a width reduction step, we know that $\sum_e \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq 2\rho \Psi(\bar{\mathbf{r}}^{(i,k)})$, and therefore combining this and the inequality above, we must have

$$\sum_{e \in H} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq \rho \Psi(\bar{\mathbf{r}}^{(i,k)}).$$

Recall that for any $\zeta = \rho, 2\rho, 4\rho, \dots, 2^{c\rho}\rho$ where $2^{c\rho}\rho \geq \sqrt{n/\epsilon}$, the algorithm defines the set $H_\zeta = \{e \in H \mid |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \in [\zeta, 2\zeta]\}$. Note that since $\Psi(\bar{\mathbf{r}}^{(i,k)}) = \sum_e \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2$, for any e we have

$$|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \leq \left(\frac{\Psi(\bar{\mathbf{r}}^{(i,k)})}{\bar{\mathbf{r}}_e^{(i,k)}} \right)^{1/2} \leq \sqrt{\frac{e^{\epsilon+2\delta} n}{\epsilon}} < 2\sqrt{\frac{n}{\epsilon}},$$

where the second step follows from $\bar{\mathbf{r}}_e^{(i,k)} \geq e^{-\delta} \frac{\epsilon}{n} \Phi(\mathbf{w}^{(i,k)}) \geq e^{-\epsilon-2\delta} \frac{\epsilon}{n} \Psi(\bar{\mathbf{r}}^{(i,k)})$ using Lemma A.1. So we have

$$\bigcup_{\zeta=\rho}^{2^{c\rho}\rho} H_\zeta = H.$$

There are at most $\log(\frac{n}{\epsilon\rho})$ such ζ 's, so this implies that there must exist a ζ^* that satisfies $\sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq \frac{\rho \Psi(\bar{\mathbf{r}}^{(i,k)})}{\log(\frac{n}{\epsilon\rho})}$, and we can find it on Line 29 in Algorithm 11.

Again using Lemma A.3 we have that

$$\begin{aligned}\Psi(\bar{\mathbf{r}}^{(i,k+1)}) &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \sum_e \left(\frac{\bar{\mathbf{r}}_e^{(i,k+1)} - \bar{\mathbf{r}}_e^{(i,k)}}{\bar{\mathbf{r}}_e^{(i,k+1)}} \right) \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \\ &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \frac{0.9\epsilon}{1+3\epsilon} \cdot \sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2,\end{aligned}\tag{10}$$

where the second step follows from when $H = S$ the width reduction step only updates edges in H_{ζ^*} and Eq. (9),

Next we prove two lower bounds of $\sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2$. On the one hand, we have

$$\sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \geq |H_{\zeta^*}| \cdot e^{-\epsilon-2\delta} \frac{\epsilon}{n} \Psi(\bar{\mathbf{r}}^{(i,k)}) \cdot (\zeta^*)^2,$$

which follows from $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \geq \zeta^*$ for all $e \in H_{\zeta^*}$, and that $\bar{\mathbf{r}}_e^{(i,k)} \geq e^{-\delta} \cdot \frac{\epsilon}{n} \Phi(\mathbf{w}^{(i,k)}) \geq e^{-\epsilon-2\delta} \cdot \frac{\epsilon}{n} \Psi(\bar{\mathbf{r}}^{(i,k)})$. On the other hand, we also have

$$\begin{aligned}\sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 &\geq \frac{1}{\max_{e \in H_{\zeta^*}} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e} \sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \\ &\geq \frac{1}{2\zeta^*} \cdot \frac{\rho \Psi(\bar{\mathbf{r}}^{(i,k)})}{\log(\frac{n}{\epsilon\rho})},\end{aligned}$$

where the second step follows from $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \leq 2\zeta^*$ for all $e \in H_{\zeta^*}$, and $\sum_{e \in H_{\zeta^*}} \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq \frac{\rho \Psi(\bar{\mathbf{r}}^{(i,k)})}{\log(\frac{n}{\epsilon\rho})}$.

Plugging these two lower bounds into Eq. (10) we have

$$\begin{aligned}\Psi(\bar{\mathbf{r}}^{(i,k+1)}) &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) \cdot \left(1 + \frac{0.9\epsilon}{1+3\epsilon} \cdot \max \left\{ e^{-\epsilon-2\delta} \cdot \frac{\epsilon \cdot |H_{\zeta^*}| \cdot (\zeta^*)^2}{n}, \frac{1}{2 \log(\frac{n}{\epsilon\rho})} \cdot \frac{\rho}{\zeta^*} \right\} \right) \\ &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) \cdot \left(1 + O \left(\epsilon^{4/3} \cdot \frac{|H_{\zeta^*}|^{1/3} \cdot \rho^{2/3}}{n^{1/3} \cdot \log^{2/3}(\frac{n}{\epsilon\rho})} \right) \right),\end{aligned}$$

where the second step follows from $a + b \geq a^{1/3}b^{2/3}$.

Finally, note that when $H \neq S$ the size of the width reduction step is $\text{SIZE}(k) = |H| + 1$, and when $H = S$ the size of the width reduction step is $\text{SIZE}(k) = |H_{\zeta^*}|$. So combining these two cases we can conclude that we always have

$$\Psi(\bar{\mathbf{r}}^{(i,k+1)}) \geq \Psi(\bar{\mathbf{r}}^{(i,k)}) \cdot \left(1 + O \left(\frac{\epsilon^{4/3} \cdot \rho^{2/3} \cdot \text{SIZE}(k)^{1/3}}{n^{1/3} \cdot \log^{2/3}(\frac{n}{\epsilon\rho})} \right) \right).$$

Primal step. Next we prove that Ψ doesn't decrease by too much after any primal step i .

Since $\mathbf{w}_e^{(i+1,k_i)} = \mathbf{w}_e^{(i,k_i)} (1 + \epsilon \vec{\alpha}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e)$, we have

$$\begin{aligned}\mathbf{r}_e^{(i+1,k_i)} &= \mathbf{w}_e^{(i+1,k_i)} + \frac{\epsilon}{n} \Phi(\mathbf{w}^{(i+1,k_i)}) \\ &\geq \mathbf{w}_e^{(i,k_i)} - \epsilon \alpha_+ \mathbf{w}_e^{(i,k_i)} |\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}|_e + \frac{\epsilon}{n} \Phi(\mathbf{w}^{(i,k_i)}) (1 - \epsilon \alpha_+ e^{\epsilon+\delta}) \\ &\geq \mathbf{w}_e^{(i,k_i)} (1 - \epsilon) + \frac{\epsilon}{n} \Phi(\mathbf{w}^{(i,k_i)}) (1 - \epsilon) = \mathbf{r}_e^{(i,k_i)} (1 - \epsilon),\end{aligned}\tag{11}$$

where the second step follows from $\vec{\alpha}^{(i,k_i)} \leq \alpha_+$ and $\Phi(\mathbf{w}^{(i+1,k_i)}) \geq \Phi(\mathbf{w}^{(i,k_i)}) \cdot (1 - \epsilon\alpha_+e^{\epsilon+\delta})$ by Lemma G.3, the third step follows from $\alpha_+|\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}| \leq 0.1$ from the same proof as that of Lemma J.1. Similarly we also have

$$\mathbf{r}_e^{(i+1,k_i)} \leq (1 + \epsilon)\mathbf{r}_e^{(i,k_i)}. \quad (12)$$

We also have,

$$\begin{aligned} \left| \frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i,k_i)}}{\mathbf{r}_e^{(i+1,k_i)}} \right| &\leq \frac{|\mathbf{w}_e^{(i+1,k_i)} - \mathbf{w}_e^{(i,k_i)}| + \frac{\epsilon}{n}|\Phi(\mathbf{w}^{(i+1,k_i)}) - \Phi(\mathbf{w}^{(i,k_i)})|}{\mathbf{r}_e^{(i,k_i)}(1 - \epsilon)} \\ &\leq \frac{\epsilon\alpha_+|\mathbf{w}_e^{(i,k_i)}| \tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}|_e + \frac{\epsilon}{n} \cdot e^{\epsilon+\delta}\epsilon\alpha_+|\Phi(\mathbf{w}^{(i,k_i)})|}{\mathbf{r}_e^{(i,k_i)}(1 - \epsilon)} \\ &\leq \frac{\epsilon\alpha_+|\mathbf{r}_e^{(i,k_i)}| \tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}|_e + e^{\epsilon+\delta}\epsilon\alpha_+|\mathbf{r}_e^{(i,k_i)}|}{\mathbf{r}_e^{(i,k_i)}(1 - \epsilon)} \\ &\leq (1 + 2\epsilon)\epsilon\alpha_+|\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}|_e + (1 + 4\epsilon)\epsilon\alpha_+. \end{aligned} \quad (13)$$

Next we consider the two cases that could happen to the coordinate e in the i -th primal step. From now on, when it's clear from the context, we will use $\mathbf{r}_e^{(i)}$ to refer to $\mathbf{r}_e^{(i,k_{i-1})}$, and similarly $\bar{\mathbf{r}}_e^{(i)}$ to refer to $\bar{\mathbf{r}}_e^{(i,k_{i-1})}$, so that this is consistent with the notations used in SELECTVECTOR.

1. If SELECTVECTOR doesn't update $\bar{\mathbf{r}}_e$ on the i -th iteration, then we have $\bar{\mathbf{r}}_e^{(i+1,k_i)} = \bar{\mathbf{r}}_e^{(i,k_i)}$.
2. If SELECTVECTOR does update $\bar{\mathbf{r}}_e$ on the i -th iteration, i.e., $e \in S_i$, then defining $j_{i,e} := \max\{\text{LASTWIDTH}(i, e), \text{LAST}(i, e)\}$, i.e., $j_{i,e} \leq i$ is the last primal iterate during which the algorithm updates \mathbf{w}_e . Define $\ell_{i,e}$ to be the smallest integer ℓ such that $i + 1 \equiv 0 \pmod{2^\ell}$ and $|\ln(\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i+1-2^\ell)}})| \geq \frac{\delta}{2\log n}$. By definition we have $\bar{\mathbf{r}}_e^{(i+1,k_i)} = \mathbf{r}_e^{(i+1,k_i)}$, and $\bar{\mathbf{r}}_e^{(i,k_i)} = \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$.

Further note that in this case we have the following properties:

- For all $j \in [j_{i,e} + 1, i]$, the value of $\bar{\mathbf{r}}_e$ remains the same for all width reduction steps between the $(j - 1)^{\text{th}}$ and j^{th} primal steps, i.e., $\bar{\mathbf{r}}_e^{(j,k_{j-1})} = \bar{\mathbf{r}}_e^{(j,k_j)}$. This is because there is no width reduction step that updates the weight of e in these iterations.
- For all $j \in [j_{i,e} + 1, i]$, $\mathbf{r}_e^{(j,k_j)} \approx_\delta \bar{\mathbf{r}}_e^{(j,k_j)} = \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$. This is because the value of $\bar{\mathbf{r}}_e$ remains the same from primal iterations $(j_{i,e} + 1)$ to i , and it is always a δ -approximation of the true value of \mathbf{r}_e .
- If $i + 1 - 2^{\ell_{i,e}} > j_{i,e}$, then

$$\begin{aligned} |\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}| &\leq \delta \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})} \\ &\leq 2\log n \cdot \left| \ln\left(\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}}\right) \right| \cdot \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})} \\ &\leq 5\log n \cdot |\mathbf{r}_e^{(i+1)} - \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}| \end{aligned}$$

where the first step follows from $\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})} \approx_\delta \bar{\mathbf{r}}_e^{(i+1-2^{\ell_{i,e}})} = \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$ since $i + 1 - 2^{\ell_{i,e}} > j_{i,e}$, the second step follows from $|\ln(\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}})| \geq \frac{\delta}{2\log n}$, the third step

follows from $|\ln(\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}})| \leq 2|\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}} - 1|$ since $|\ln(\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}})| \leq 0.1$, and this is because $\mathbf{r}_e^{(i,k_i)} \approx_{\delta} \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})} \approx_{\delta} \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}$ by the second bullet and $\mathbf{r}_e^{(i+1,k_i)} \approx_{\epsilon} \mathbf{r}_e^{(i,k_i)}$ by Eq. (11) and (12).

Because of the third bullet point above, if $j_{i,e} < i + 1 - 2^{\ell_{i,e}}$ then we have

$$\begin{aligned} |\bar{\mathbf{r}}_e^{(i+1,k_i)} - \bar{\mathbf{r}}_e^{(i,k_i)}| &= |\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}| \\ &\leq |\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}| + |\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}| \\ &\leq 10 \log n \cdot |\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}|. \end{aligned}$$

From now on we can without loss of generality assume that $j_{i,e} \geq i + 1 - 2^{\ell_{i,e}}$, since otherwise we can upper bound $|\bar{\mathbf{r}}_e^{(i+1,k_i)} - \bar{\mathbf{r}}_e^{(i,k_i)}|$ by $\tilde{O}(|\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}|)$ instead of $|\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}|$. Now,

$$\begin{aligned} \Psi(\bar{\mathbf{r}}^{(i+1,k_i)}) &\stackrel{(i)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \sum_e \left| \frac{\bar{\mathbf{r}}_e^{(i+1,k_i)} - \bar{\mathbf{r}}_e^{(i,k_i)}}{\bar{\mathbf{r}}_e^{(i+1,k_i)}} \right| \bar{\mathbf{r}}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \\ &\stackrel{(ii)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \sum_{e \in S_i} \left| \frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}}{\mathbf{r}_e^{(i+1,k_i)}} \right| \bar{\mathbf{r}}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \\ &\stackrel{(iii)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \sum_{e \in S_i} \left| \frac{\sum_{j=j_{i,e}}^i (\mathbf{r}_e^{(j+1,k_j)} - \mathbf{r}_e^{(j,k_j)})}{\mathbf{r}_e^{(i+1,k_i)}} \right| \bar{\mathbf{r}}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \\ &\stackrel{(iv)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \frac{e^{2\delta}}{(1-\epsilon)} \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \left| \frac{(\mathbf{r}_e^{(j+1,k_j)} - \mathbf{r}_e^{(j,k_j)})}{\mathbf{r}_e^{(j+1,k_j)}} \right| \bar{\mathbf{r}}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \\ &\stackrel{(v)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - O(1) \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \left(\epsilon \alpha |\tilde{\mathbf{C}}\Delta^{(j,k_j)} - \tilde{\mathbf{d}}|_e + \epsilon \alpha \right) \bar{\mathbf{r}}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \\ &\stackrel{(vi)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - O(1) \cdot \epsilon \alpha \cdot \sum_{e \in S_i} (i + 1 - j_{i,e}) \cdot \bar{\mathbf{r}}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \\ &\quad - O(1) \cdot \epsilon \alpha \cdot \sum_{e \in S_i} (i + 1 - j_{i,e}) \cdot \bar{\mathbf{r}}_e^{(i,k_i)} |\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}|_e^3 \\ &\quad - O(1) \cdot \epsilon \alpha \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^{i-1} \bar{\mathbf{r}}_e^{(j,k_j)} |\tilde{\mathbf{C}}\Delta^{(j,k_j)} - \tilde{\mathbf{d}}|_e^3, \end{aligned}$$

where (i) follows from Lemma A.3, (ii) follows from $\bar{\mathbf{r}}_e^{(i+1,k_i)} = \mathbf{r}_e^{(i+1,k_i)}$, and $\bar{\mathbf{r}}_e^{(i,k_i)} = \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$ for $e \in S_i$, (iv) follows from $\mathbf{r}_e^{(j+1,k_j)} \approx_{\delta} \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$ and $\mathbf{r}_e^{(i,k_i)} \approx_{\delta} \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$ as we argued above, and since we also have $\mathbf{r}_e^{(i+1,k_i)} \geq (1-\epsilon)\mathbf{r}_e^{(i,k_i)}$ from Eq. (11), combining these we have $\mathbf{r}_e^{(j+1,k_j)} \leq e^{\delta} \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})} \leq e^{2\delta} \mathbf{r}_e^{(i,k_i)} \leq \frac{e^{2\delta}}{(1-\epsilon)} \mathbf{r}_e^{(i+1,k_i)}$. Step (v) follows from Eq. (13), (vi) follows from AM-GM inequality that $|\tilde{\mathbf{C}}\Delta^{(j,k_j)} - \tilde{\mathbf{d}}|_e (\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \leq \frac{1}{3} \cdot |\tilde{\mathbf{C}}\Delta^{(j,k_j)} - \tilde{\mathbf{d}}|_e^3 + \frac{2}{3} \cdot |\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}|_e^3$ and that $\mathbf{r}_e^{(j,k_j)} \approx_{\delta} \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})} \approx_{\delta} \bar{\mathbf{r}}_e^{(i,k_i)}$ for all $e \in S_i$ and $j \in [j_{i,e}, i]$.

With an abuse of notation, let ℓ_i denote the largest integer such that $i + 1 \equiv 0 \pmod{2^{\ell_i}}$. Since we assumed that $j_{i,e} \geq i + 1 - 2^{\ell_{i,e}}$ for all $e \in S_i$, we have $i + 1 - j_{i,e} \leq 2^{\ell_{i,e}} \leq 2^{\ell_i}$. Also

note that in primal steps we have $\sum_e \bar{r}_e^{(i,k)} |\tilde{C}\Delta^{(i,k)} - \tilde{d}|_e^3 \leq 2\rho\Psi(\bar{r}^{(i,k)})$, so the above equation becomes, for some $C_1 = \tilde{O}(1)$

$$\Psi(\bar{r}^{(i+1,k)}) \geq \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot 2^{\ell_i}\right) \cdot \Psi(\bar{r}^{(i,k)}) - C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^{i-1} \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3. \quad (14)$$

Recall that we defined $L = \Theta(\frac{1}{(\log^4 n) \cdot \epsilon\alpha\rho})$ to be a power of 2. Next we consider the iterations between $(c-1)L$ and cL using Eq. (14). We have

$$\begin{aligned} & \Psi(\bar{r}^{(cL, k_{cL-1})}) \\ & \geq \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot 2^{\ell_{cL-1}}\right) \cdot \Psi(\bar{r}^{(cL-1, k_{cL-1})}) - C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_{cL-1}} \sum_{j=j_{cL-1,e}}^{cL-2} \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3 \\ & \geq \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot (2^{\ell_{cL-1}} + 2^{\ell_{cL-2}})\right) \cdot \frac{\Psi(\bar{r}^{(cL-1, k_{cL-1})})}{\Psi(\bar{r}^{(cL-1, k_{cL-2})})} \cdot \Psi(\bar{r}^{cL-2, k_{cL-2}}) \\ & \quad - \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot 2^{\ell_{cL-1}}\right) \cdot \frac{\Psi(\bar{r}^{(cL-1, k_{cL-1})})}{\Psi(\bar{r}^{(cL-1, k_{cL-2})})} \cdot C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_{cL-2}} \sum_{j=j_{cL-2,e}}^{cL-3} \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3 \\ & \quad - C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_{cL-1}} \sum_{j=j_{cL-1,e}}^{cL-2} \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3 \\ & \geq \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot (2^{\ell_{cL-1}} + 2^{\ell_{cL-2}} + 2)\right) \cdot \frac{\Psi(\bar{r}^{(cL-1, k_{cL-1})})}{\Psi(\bar{r}^{(cL-1, k_{cL-2})})} \cdot \Psi(\bar{r}^{cL-2, k_{cL-2}}) \\ & \quad - \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot 2^{\ell_{cL-1}}\right) \cdot \frac{\Psi(\bar{r}^{(cL-1, k_{cL-1})})}{\Psi(\bar{r}^{(cL-1, k_{cL-2})})} \cdot C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_{cL-2}} \sum_{j=j_{cL-2,e}}^{cL-3} \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3 \\ & \quad - C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_{cL-1}} \sum_{j=j_{cL-1,e}}^{cL-3} \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3 \\ & \geq \dots \\ & \geq \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot (2L + \sum_{i=(c-1)L+1}^{cL} 2^{\ell_{i-1}})\right) \cdot \prod_{i=(c-1)L}^{cL} \frac{\Psi(\bar{r}^{(i-1, k_{i-1})})}{\Psi(\bar{r}^{(i-1, k_{i-2})})} \cdot \Psi(\bar{r}^{(c-1)L, k_{(c-1)L}}), \end{aligned}$$

where the second step follows from $\Psi(\bar{r}^{(cL-1, k_{cL-2})}) \geq \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot 2^{\ell_{cL-2}}\right) \cdot \Psi(\bar{r}^{cL-2, k_{cL-2}}) - C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_{cL-2}} \sum_{j=j_{cL-2,e}}^{cL-3} \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3$, the third step follows from taking out all terms $\bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3$ for $j = cL - 2$ and that $\sum_e \bar{r}_e^{(j,k_j)} |\tilde{C}\Delta^{(j,k_j)} - \tilde{d}|_e^3 \leq 2\rho\Psi(\bar{r}^{(j,k_j)})$, and the last two steps follow from repeat this process for L times, and noting that for any $cL - t$, a coordinate e can only be in one $S_{cL-t'}$ where $t' < t$ and $j_{cL-t',e} \leq cL - t$, and also noting that we proved that we can wlog assume $j_{cL-t,e} \geq cL - t + 1 - 2^{\ell_{cL-t,e}}$ for all $t \leq L$, and this is then $\geq (c-1)L$ since L is a power of 2.

Finally, note that $\prod_{i=(c-1)L}^{cL} \frac{\Psi(\bar{r}^{(i-1, k_{i-1})})}{\Psi(\bar{r}^{(i-1, k_{i-2})})}$ is exactly the increase that we get from the width reduction steps, and also note that we have $\sum_{i=(c-1)L+1}^{cL} 2^{\ell_{i-1}} \leq L \cdot \log n$ since by definition ℓ_{i-1} is the largest integer ℓ such that $i \equiv 0 \pmod{2^\ell}$. And this gives the claimed lower bound of this lemma. \square

G.2.4 Putting Everything Together: Analysis of Algorithm

Next we analyze the iteration complexity and the error of Algorithm 11. We first bound the total number of width reduction steps. In the following lemma we denote the hidden factors in Lemma G.5 as $C_2 \leq O(\log^3 n)$ and $C_3 \geq O(1)$ such that

$$\Psi(\bar{\mathbf{r}}^{(cL, k_{cL})}) \geq \Psi(\bar{\mathbf{r}}^{((c-1)L, k_{(c-1)L})}) \cdot \left(1 - C_2 \epsilon \alpha \rho L\right) \cdot \prod_{k=k_{(c-1)L}}^{k_{cL}} \left(1 + C_3 \frac{\epsilon^{4/3} \rho^{2/3} \cdot \text{SIZE}(k)^{1/3}}{n^{1/3} \cdot \log^{2/3}(\frac{n}{\epsilon \rho})}\right).$$

Lemma G.6 (Number of width reduction steps). *The total number of width reduction steps of Algorithm 11 is at most $O\left(\frac{n^{1/3} \rho^{1/3} \log^5 n}{\epsilon^{10/3}} \cdot \log(\frac{n}{\Psi_0})\right)$ for large enough n .*

Proof. Let K denote the total number of width reduction steps. We first assume that we halt the algorithm if there are more than $K' = 10^4 C_3^{-1} \cdot \frac{n^{1/3} \rho^{1/3} \log^6 n}{\epsilon^{10/3}} \cdot \log(\frac{n}{\Psi_0})$ width reduction steps. We will then prove that $K \leq 9000 C_3^{-1} \cdot \frac{n^{1/3} \rho^{1/3} \log^5 n}{\epsilon^{10/3}} \cdot \log(\frac{n}{\Psi_0}) < K'$, which means we can make this assumption without changing the algorithm. Under this assumption, and using Lemma G.3, we have that during the algorithm, we always have

$$\begin{aligned} \Phi(\mathbf{w}^{(i,k)}) &\leq \Phi(\mathbf{w}^{(0,0)}) \cdot \left(1 + \epsilon \alpha e^{\epsilon+\delta}\right)^T \cdot \left(1 + \epsilon e^{\epsilon+2\delta}(\tau^{-1} + \rho^{-2})\right)^{K'} \\ &\leq n \cdot \exp(2\epsilon \alpha \cdot \alpha^{-1} \epsilon^{-2} \log n) \cdot \exp(2\epsilon(\tau^{-1} + \rho^{-2}) 10^4 C_3^{-1} \cdot \frac{n^{1/3} \rho^{1/3} \log^6 n}{\epsilon^{10/3}} \cdot \log(\frac{n}{\Psi_0})) \\ &\leq n^{2/\epsilon + 10^4 C_3^{-1}/\epsilon} \leq n^{3 \log n / \epsilon}, \end{aligned} \quad (15)$$

where in the second and third steps we used the parameters of Algorithm 11 that $T = \alpha^{-1} \epsilon^{-2} \log n$, $\tau = n^{1/2-\eta} \cdot \epsilon^{-4} \cdot \log(n)^8 \log(\frac{n}{\Psi_0})^2$, $\rho = n^{1/2-3\eta} \cdot \epsilon^{-2} \cdot \log(n)^4 \log(\frac{n}{\Psi_0})$, and $\eta \leq 1/10$, in the last step we assume n is large enough such that $n \geq 10^4 C_3^{-1}$.

Consider any integer $c \geq 1$. We next bound the number of width reduction steps between primal steps $(c-1)L$ and cL , and in this proof we denote this number as K_c .

Using Lemma G.5, we have

$$\begin{aligned} \left(1 + C_3 \frac{\epsilon^{4/3} \cdot \rho^{2/3}}{n^{1/3} \log^{2/3}(\frac{n}{\epsilon \rho})}\right)^{K_c} &\leq \frac{2\Psi(\bar{\mathbf{r}}^{(cL, k_{cL})})}{\Psi(\bar{\mathbf{r}}^{((c-1)L, k_{(c-1)L})})} \leq \frac{4\Psi(\bar{\mathbf{r}}^{(cL, k_{cL})})}{\Psi_0} \leq \frac{10 \cdot n^{3 \log n / \epsilon}}{\Psi_0} \\ \Rightarrow K_c &\leq \frac{3 \cdot n^{1/3} \log n}{C_3 \cdot \epsilon^{4/3} \cdot \rho^{2/3}} \cdot \frac{3 \log n}{\epsilon} \cdot \log(\frac{n}{\Psi_0}), \end{aligned}$$

where the first step follows from Lemma G.5 and that we always have $\text{SIZE}(k) \geq 1$ for all k , and that the $C_2 \epsilon \alpha \rho \cdot L$ factor of Lemma G.5 is upper bounded by $1/2$ since $L \leq \frac{1}{100(\log^4 n) \epsilon \alpha \rho}$, the second step follows from the same proof as Lemma J.4 that $\Psi(\bar{\mathbf{r}}^{((c-1)L, k_{(c-1)L})}) \geq \frac{1}{1+2\epsilon} \cdot \Psi(\bar{\mathbf{r}}^{(0,0)})$ and Lemma A.2 that $\Psi(\bar{\mathbf{r}}^{(0,0)}) \geq \Psi_0$, and the third step follows from $\Psi(\bar{\mathbf{r}}^{(cL, k_{cL})}) \leq e^{\epsilon+\delta} \Phi(\mathbf{w}^{(cL, k_{cL})}) \leq e^{\epsilon+\delta} n^{3 \log n / \epsilon}$ by Lemma A.1 and Eq. (15).

Note that the above bound holds for any integer c . Since there are in total $T = \alpha^{-1} \epsilon^{-2} \log n$ number of primal steps, and since $L \leq \frac{1}{100 \epsilon \alpha \rho \log^4 n}$, the total number of width reduction steps is upper bounded by

$$\begin{aligned} \frac{T}{L} \cdot K_c &\leq \frac{1000 \rho \log^3 n}{\epsilon} \cdot \frac{3 \cdot n^{1/3} \log n}{C_3 \cdot \epsilon^{4/3} \cdot \rho^{2/3}} \cdot \frac{3 \log n}{\epsilon} \cdot \log(\frac{n}{\Psi_0}) \\ &\leq 9000 C_3^{-1} \cdot \frac{n^{1/3} \rho^{1/3} \log^5 n}{\epsilon^{10/3}} \cdot \log(\frac{n}{\Psi_0}). \end{aligned} \quad \square$$

Next we bound the error of Algorithm 11.

Lemma G.7 (Error of Algorithm 11). *Algorithm 11 outputs a vector $\hat{\mathbf{x}} \in \mathbb{R}^d$ such that $\|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty \leq 1 + O(\epsilon)$.*

Proof. First note that all the requirements on the parameters of the lemmas in Section G.2 are satisfied by the parameters of Algorithm 11 for $\eta \leq 1/10$.

Let $\hat{\mathbf{x}} = \frac{\mathbf{x}}{T}$ be the solution returned by Algorithm 11. We will bound the objective value at $\hat{\mathbf{x}}$. The algorithm has $T = \alpha^{-1}\epsilon^{-2}\log n$ primal steps, and by Lemma G.6 we know that it has at most $K \leq O\left(\frac{n^{1/3}\rho^{1/3}\log^5 n}{\epsilon^{10/3}} \cdot \log\left(\frac{n}{\Psi_0}\right)\right)$ width reduction steps. We can now apply Lemma G.3 to get,

$$\Phi(\mathbf{w}^{(T,K)}) \leq n \cdot e^{O(\epsilon\alpha T + \epsilon(\tau^{-1} + \rho^{-2})K)} \leq n^{O(\frac{1}{\epsilon})},$$

where the second step follows from the parameters of Algorithm 11 that $\alpha = n^{-1/2+\eta} \cdot \epsilon \cdot \log(n)^{-4/3} \log(\frac{n}{\Psi_0})^{-1/3}/10$, $\tau = n^{1/2-\eta} \cdot \epsilon^{-4} \cdot \log(n)^8 \log(\frac{n}{\Psi_0})^2$, and $\rho = n^{1/2-3\eta} \cdot \epsilon^{-2} \cdot \log(n)^4 \log(\frac{n}{\Psi_0})$.

We bound the ℓ_∞ norm of $\mathbf{C}\hat{\mathbf{x}} - \mathbf{d} = \frac{1}{T} \cdot \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})$ using the upper bound of the potential. Since $\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} \\ -\mathbf{C} \end{bmatrix}$, and $\tilde{\mathbf{d}} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix}$, we have that the weights $\mathbf{w} \in \mathbb{R}^{2n}$.

Therefore, for $\mathbf{w}_+ \in \mathbb{R}^n$ and $\mathbf{w}_- \in \mathbb{R}^n$, we can write $\mathbf{w}^{(i,k)} = \begin{bmatrix} \mathbf{w}_+^{(i,k)} \\ \mathbf{w}_-^{(i,k)} \end{bmatrix}$, and we have that

$\Phi(\mathbf{w}) = \sum_{e \in [n]} \mathbf{w}_{+e} + \mathbf{w}_{-e}$. We can similarly define $\bar{\mathbf{r}}_+$ and $\bar{\mathbf{r}}_-$ such that $\bar{\mathbf{r}} = \begin{bmatrix} \bar{\mathbf{r}}_+ \\ \bar{\mathbf{r}}_- \end{bmatrix}$. Since $\Delta^{(i,k)}$ is obtained by solving,

$$\Delta^{(i,k)} = \arg \min_{\Delta} \sum_{e \in [2n]} \bar{\mathbf{r}}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e^2 = \sum_{e \in [n]} (\bar{\mathbf{r}}_+^{(i,k)} + \bar{\mathbf{r}}_-^{(i,k)})_e (\mathbf{C}\Delta - \mathbf{d})_e^2,$$

the update rule $\mathbf{w}^{(i+1,k)} = \mathbf{w}^{(i,k)} \cdot (1 + \epsilon \vec{\alpha}^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}))$ implies that in every primal step,

$$\begin{aligned} (\mathbf{w}_+)_e^{(i+1,k)} &= (\mathbf{w}_+)_e^{(i,k)} \cdot (1 + \epsilon \vec{\alpha}^{(i,k)} (\mathbf{C}\Delta^{(i,k)} - \mathbf{d})_e), \\ (\mathbf{w}_-)_e^{(i+1,k)} &= (\mathbf{w}_-)_e^{(i,k)} \cdot (1 - \epsilon \vec{\alpha}^{(i,k)} (\mathbf{C}\Delta^{(i,k)} - \mathbf{d})_e). \end{aligned}$$

Now,

$$\begin{aligned} (\mathbf{w}_+)_e^{(T,K)} &= (\mathbf{w}_+)_e^{(0,0)} \cdot \prod_{i=0}^{T-1} \left(1 + \epsilon \vec{\alpha}_e^{(i,k_i)} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e\right) \\ &= \prod_{i: (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \geq 0} (1 + \epsilon \alpha_+ (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e) \cdot \prod_{i: (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e < 0} (1 + \epsilon \alpha_- (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e) \\ &\geq \exp\left(\epsilon(1 - \epsilon)\alpha \cdot \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e\right), \end{aligned}$$

where the second step follows from $\mathbf{w}_+^{(0,0)} = \mathbf{1}_n$, and $\vec{\alpha}_e^{(i,k_i)} = \alpha_+$ if $(\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \geq 0$ and $\vec{\alpha}_e^{(i,k_i)} = \alpha_-$ otherwise, the third step follows from $1 + \epsilon x \geq \exp(\epsilon(1 - \epsilon)x)$ for all $0 \leq x \leq 1$ and $1 + \epsilon x \geq \exp(\epsilon(1 + \epsilon)x)$ for all $-1 \leq x \leq 0$, and we have that $|\vec{\alpha}^{(i,k)} \cdot (\mathbf{C}\Delta^{(i,k)} - \mathbf{d})| \leq \frac{1}{10}$ by Lemma J.1. Similarly, we also get,

$$(\mathbf{w}_-)_e^{(T,K)} \geq \exp\left(\epsilon(1 - \epsilon)\alpha \cdot \sum_{i=0}^{T-1} -(\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e\right).$$

This implies that

$$\left| \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \right| \leq \frac{\ln((\mathbf{w}_+)_e^{(T,K)} + (\mathbf{w}_-)_e^{(T,K)})}{\epsilon(1-\epsilon)\alpha} \leq \frac{\ln(\Phi(\mathbf{w}^{(T,K)}))}{\epsilon(1-\epsilon)\alpha}. \quad (16)$$

So we have

$$\begin{aligned} \|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty &= \frac{1}{T} \max_e \left| \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \right| \\ &\leq \frac{\ln(\Phi(\mathbf{w}^{(T,K)}))}{\alpha T} \\ &\leq \frac{\ln n + (1+\epsilon)\epsilon\alpha T + (1+\epsilon)}{\epsilon(1-\epsilon)\alpha T} \\ &\leq 1 + 10\epsilon. \end{aligned} \quad \square$$

G.3 Guarantees of Algorithm 3: Robust Primal Step

In this section, we finally present the analysis of Algorithm 3 which additionally approximates the primal steps via a *sketch*. We will again use the two potentials as defined in Eq. (3) and (4).

In the next section, we will first present the properties of the sketching matrices that we need to use. This constitutes the major part of the section. The remaining would directly build on the analysis of the previous section.

G.3.1 Sketching Bounds

Lemma G.8 (Coordinate-wise embedding, Lemma E.5 of [LSZ19]). *Let $\mathbf{S} \in \mathbb{R}^{b \times n}$ be sampled from distribution Π such that each entry is $+\frac{1}{\sqrt{b}}$ with probability $1/2$ and $-\frac{1}{\sqrt{b}}$ with probability $1/2$. For any fixed vectors $\mathbf{g}, \mathbf{h} \in \mathbb{R}^n$, the following properties hold:*

1. $\mathbb{E}_{\mathbf{S} \sim \Pi} [\mathbf{g}^\top \mathbf{S}^\top \mathbf{S} \mathbf{h}] = \mathbf{g}^\top \mathbf{h},$
2. $\mathbb{E}_{\mathbf{S} \sim \Pi} [(\mathbf{g}^\top \mathbf{S}^\top \mathbf{S} \mathbf{h})^2] \leq (\mathbf{g}^\top \mathbf{h})^2 + \frac{C_1}{b} \|\mathbf{g}\|_2^2 \|\mathbf{h}\|_2^2,$
3. $\Pr_{\mathbf{S} \sim \Pi} \left[|\mathbf{g}^\top \mathbf{S}^\top \mathbf{S} \mathbf{h} - \mathbf{g}^\top \mathbf{h}| \leq \frac{C_2}{\sqrt{b}} \|\mathbf{g}\|_2 \|\mathbf{h}\|_2 \right] \geq 1 - 1/n^4.$

where $C_1 = O(1)$, $C_2 = O(\log n)$.

Lemma G.9 (Bounds for the vector $\hat{\mathbf{u}}$). *For all $i \in [0 : T]$, the vector $\hat{\mathbf{u}}^{(i,k)} = (\overline{\mathbf{R}}^{(i,k)})^{-1/2} \cdot (\mathbf{S}^{(i)})^\top \mathbf{S}^{(i)} \cdot (\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}$ satisfies the following properties:*

1. **Expectation.** $\mathbb{E}_{\mathbf{S}^{(i)}} [\hat{\mathbf{u}}^{(i,k)} \mid \mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}] = \mathbf{u}^{(i,k)}.$
2. **Variance.** For any vector $\mathbf{g} \in \mathbb{R}^n$ that is independent of $\mathbf{S}^{(i)}$,

$$\text{Var}_{\mathbf{S}^{(i)}} \left[\sum_e \mathbf{g}_e \hat{\mathbf{u}}_e^{(i,k)} \mid \mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)} \right] \leq \frac{C_1}{b} \cdot \mathbf{g}^\top (\overline{\mathbf{R}}^{(i,k)})^{-1} \mathbf{g} \cdot (\mathbf{u}^{(i,k)})^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{u}^{(i,k)}.$$

In particular, if $\mathbf{w}^{(i,k)} \geq 0$, then this implies that:

$$\bullet \text{ Var}_{\mathbf{S}^{(i)}} \left[\hat{\mathbf{u}}_e^{(i,k)} \mid \mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)} \right] \leq \frac{C_1 n}{\epsilon b}.$$

- $\mathbb{E}_{\mathbf{S}^{(i)}}[\widehat{\Psi}(\bar{\mathbf{r}}^{(i,k)}, \mathbf{S}^{(i)}) \mid \mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}] \leq (1 + \frac{C_1 \cdot n}{b}) \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})$.
- $\text{Var}_{\mathbf{S}^{(i)}}[\sum_e \mathbf{w}_e^{(i,k)} \widehat{\mathbf{u}}_e^{(i,k)} \mid \mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}] \leq \frac{C_1}{b} \cdot \Phi(\mathbf{w}^{(i,k)}) \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})$.

3. **Coordinate-wise absolute value.** For any vector $\mathbf{g} \in \mathbb{R}^n$ that is independent of $\mathbf{S}^{(i)}$, and when conditioned on any $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$, we have

$$\Pr_{\mathbf{S}^{(i)}} \left[\left| \sum_e \mathbf{g}_e (\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}) \right| \leq \frac{C_2}{\sqrt{b}} \cdot (\mathbf{g}^\top (\bar{\mathbf{R}}^{(i,k)})^{-1} \mathbf{g})^{1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \right] \geq 1 - 1/n^4.$$

In particular, this implies that when conditioned on any $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$:

- For any $e \in [n]$, if $\mathbf{w}_e^{(i,k)} \geq 0$, then

$$\Pr_{\mathbf{S}^{(i)}} \left[\left| \widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)} \right| \leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}} \cdot \Phi(\mathbf{w}^{(i,k)})^{-1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \right] \geq 1 - 1/n^4.$$

- If $\mathbf{w}^{(i,k)} \geq 0$, then

$$\begin{aligned} \Pr_{\mathbf{S}^{(i)}} \left[\left| \sum_e \mathbf{w}_e^{(i,k)} \cdot (\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}) \right| \leq \frac{C_2}{\sqrt{b}} \cdot \Phi(\mathbf{w}^{(i,k)})^{1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \right] &\geq 1 - 1/n^4, \\ \Pr_{\mathbf{S}^{(i)}} \left[\left| \sum_e \bar{\mathbf{r}}_e^{(i,k)} \cdot (\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}) \right| \leq \frac{C_2(1+\epsilon)}{\sqrt{b}} \cdot \Phi(\mathbf{w}^{(i,k)})^{1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \right] &\geq 1 - 1/n^4. \end{aligned}$$

4. **Symmetry.** When conditioned on any $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$, for any $i \in [T]$ and any $e \in [n]$, the distribution of $\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}$ is symmetric around zero, i.e., for any $z \in \mathbb{R}$,

$$\Pr_{\mathbf{S}^{(i)}}[\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)} = z] = \Pr_{\mathbf{S}^{(i)}}[\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)} = -z].$$

Proof. In this proof we will use the following fact from Algorithm 3: for any $i \in [0 : T]$, $\mathbf{u}^{(i,k)}$, $\mathbf{w}^{(i,k)}$, and $\bar{\mathbf{r}}^{(i,k)}$ are random variables that depend on $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$, and $\widehat{\mathbf{u}}^{(i,k)}$ is a random variable that depends on $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$, and $\mathbf{S}^{(i)}$.

Part 1 (Expectation). For any fixed $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$, we have

$$\begin{aligned} \mathbb{E}_{\mathbf{S}^{(i)}}[\widehat{\mathbf{u}}^{(i,k)}] &= \mathbb{E}_{\mathbf{S}^{(i)}}[(\bar{\mathbf{R}}^{(i,k)})^{-1/2} \cdot (\mathbf{S}^{(i)})^\top \mathbf{S}^{(i)} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}] \\ &= (\bar{\mathbf{R}}^{(i,k)})^{-1/2} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)} \\ &= \mathbf{u}^{(i,k)}, \end{aligned}$$

where the second step follows from Part 1 of Lemma G.8.

Part 2 (Variance). Consider any fixed $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$. For any vector $\mathbf{g} \in \mathbb{R}^n$ that is independent of $\mathbf{S}^{(i)}$, we have

$$\text{Var}_{\mathbf{S}^{(i)}}[\sum_e \mathbf{g}_e \widehat{\mathbf{u}}_e^{(i,k)}] = \mathbb{E}_{\mathbf{S}^{(i)}}[(\sum_e \mathbf{g}_e \widehat{\mathbf{u}}_e^{(i,k)})^2] - (\sum_e \mathbf{g}_e \mathbf{u}_e^{(i,k)})^2.$$

We also have

$$\begin{aligned} \mathbb{E}_{\mathbf{S}^{(i)}}[(\sum_e \mathbf{g}_e \widehat{\mathbf{u}}_e^{(i,k)})^2] &= \mathbb{E}_{\mathbf{S}^{(i)}}[(\mathbf{g}^\top \widehat{\mathbf{u}}^{(i,k)})^2] \\ &= \mathbb{E}_{\mathbf{S}^{(i)}} \left[\left(\mathbf{g}^\top (\bar{\mathbf{R}}^{(i,k)})^{-1/2} \cdot (\mathbf{S}^{(i)})^\top \mathbf{S}^{(i)} \cdot (\bar{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)} \right)^2 \right] \\ &\leq (\mathbf{g}^\top \mathbf{u}^{(i,k)})^2 + \frac{C_1}{b} \cdot \mathbf{g}^\top (\bar{\mathbf{R}}^{(i,k)})^{-1} \mathbf{g} \cdot (\mathbf{u}^{(i,k)})^\top \bar{\mathbf{R}}^{(i,k)} \mathbf{u}^{(i,k)}, \end{aligned}$$

where the third step follows from Part 2 of Lemma G.8.

So we have

$$\mathbf{Var}_{\mathbf{S}^{(i)}}\left[\sum_e \mathbf{g}_e \widehat{\mathbf{u}}_e^{(i,k)}\right] \leq \frac{C_1}{b} \cdot \mathbf{g}^\top (\overline{\mathbf{R}}^{(i,k)})^{-1} \mathbf{g} \cdot (\mathbf{u}^{(i,k)})^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{u}^{(i,k)}.$$

In particular, if $\mathbf{w}^{(i,k)} \geq 0$, then this implies the following bounds:

- For any $e \in [n]$, let $\mathbf{g} = \mathbf{1}_e$, then we have

$$\mathbf{Var}_{\mathbf{S}^{(i)}}\left[\widehat{\mathbf{u}}_e^{(i,k)}\right] \leq \frac{C_1}{b} \cdot \frac{\Psi(\overline{\mathbf{r}}^{(i,k)})}{\overline{\mathbf{r}}_e^{(i,k)}} \leq \frac{C_1 n}{\epsilon b},$$

where the second step follows from $(1 + \delta)\overline{\mathbf{r}} \geq \mathbf{r}^{(i,k)} = \mathbf{w}^{(i,k)} + \frac{\epsilon}{n}\Phi(\mathbf{w}^{(i,k)}) \geq \frac{\epsilon}{n}\Phi(\mathbf{w}^{(i,k)})$ when $\mathbf{w}^{(i,k)} \geq 0$.

- Let $\mathbf{g} = \mathbf{w}^{(i,k)}$. Then we have

$$\begin{aligned} \mathbf{Var}_{\mathbf{S}^{(i)}}\left[\sum_e \mathbf{w}_e^{(i,k)} \widehat{\mathbf{u}}_e^{(i,k)}\right] &\leq \frac{C_1}{b} \cdot (\mathbf{w}^{(i,k)})^\top (\overline{\mathbf{R}}^{(i,k)})^{-1} \mathbf{w}^{(i,k)} \cdot (\mathbf{u}^{(i,k)})^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{u}^{(i,k)} \\ &\leq \frac{C_1}{b} \cdot \left(\sum_e \frac{(\mathbf{w}_e^{(i,k)})^2}{\overline{\mathbf{r}}_e^{(i,k)}}\right) \cdot \Psi(\overline{\mathbf{r}}^{(i,k)}) \\ &\leq \frac{C_1}{b} \cdot \Phi(\mathbf{w}^{(i,k)}) \cdot \Psi(\overline{\mathbf{r}}^{(i,k)}), \end{aligned}$$

where the second step follows from $\Psi(\overline{\mathbf{r}}^{(i,k)}) = (\mathbf{u}^{(i,k)})^\top \overline{\mathbf{R}}^{(i,k)} \mathbf{u}^{(i,k)}$, the third step follows from $(1 + \delta)\overline{\mathbf{r}} \geq \mathbf{r}^{(i,k)} = \mathbf{w}^{(i,k)} + \frac{\epsilon}{n}\Phi(\mathbf{w}^{(i,k)}) \geq \mathbf{w}^{(i,k)}$ when $\mathbf{w}^{(i,k)} \geq 0$, and $\sum_e \mathbf{w}_e^{(i,k)} = \Phi(\mathbf{w}^{(i,k)})$.

Part 3 (Coordinate-wise absolute value). Consider any fixed $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$. For any $\mathbf{g} \in \mathbb{R}^n$ that is independent of $\mathbf{S}^{(i)}$, using Part 3 of Lemma G.8 we have that with probability at least $1 - 1/n^4$ over the randomness of $\mathbf{S}^{(i)}$,

$$\begin{aligned} \left| \sum_e \mathbf{g}_e (\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}) \right| &= \left| \mathbf{g}^\top (\overline{\mathbf{R}}^{(i,k)})^{-1/2} \cdot \left((\mathbf{S}^{(i)})^\top \mathbf{S}^{(i)} \cdot (\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)} - (\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)} \right) \right| \\ &\leq \frac{C_2}{\sqrt{b}} \|(\overline{\mathbf{R}}^{(i,k)})^{-1/2} \mathbf{g}\|_2 \cdot \|(\overline{\mathbf{R}}^{(i,k)})^{1/2} \mathbf{u}^{(i,k)}\|_2 \\ &= \frac{C_2}{\sqrt{b}} \cdot (\mathbf{g}^\top (\overline{\mathbf{R}}^{(i,k)})^{-1} \mathbf{g})^{1/2} \cdot \Psi(\overline{\mathbf{r}}^{(i,k)})^{1/2}, \end{aligned}$$

where the third step follows from $\Psi(\overline{\mathbf{r}}^{(i,k)}) = \sum_{e'} \overline{\mathbf{r}}_{e'}^{(i,k)} (\mathbf{u}_{e'}^{(i,k)})^2$.

In particular, if $\mathbf{w}^{(i,k)} \geq 0$, then this implies the following bounds:

- For any $e \in [n]$, let $\mathbf{g} = \mathbf{1}_e$, we have that with probability at least $1 - 1/n^4$ over the randomness of $\mathbf{S}^{(i)}$,

$$\begin{aligned} \left| \widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)} \right| &\leq \frac{C_2}{\sqrt{b}} (\overline{\mathbf{r}}_e^{(i,k)})^{-1/2} \cdot \Psi(\overline{\mathbf{r}}^{(i,k)})^{1/2} \\ &\leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}} \cdot \Phi(\overline{\mathbf{r}}^{(i,k)})^{-1/2} \cdot \Psi(\overline{\mathbf{r}}^{(i,k)})^{1/2} \end{aligned}$$

where the second step follows from $(1 + \delta)\overline{\mathbf{r}} \geq \mathbf{r}_e^{(i,k)} = \mathbf{w}_e^{(i,k)} + \frac{\epsilon}{n}\Phi(\mathbf{w}^{(i,k)}) \geq \frac{\epsilon}{n}\Phi(\mathbf{w}^{(i,k)})$ since $\mathbf{w}_e^{(i,k)} \geq 0$.

- Let $\mathbf{g} = \mathbf{w}^{(i,k)}$, we have that with probability at least $1 - 1/n^4$ over the randomness of $\mathbf{S}^{(i)}$,

$$\begin{aligned} \left| \sum_e \mathbf{w}_e^{(i,k)} \cdot (\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}) \right| &\leq \frac{C_2}{\sqrt{b}} \left(\sum_e \frac{(\mathbf{w}_e^{(i,k)})^2}{\bar{\mathbf{r}}_e^{(i,k)}} \right)^{1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \\ &\leq \frac{C_2}{\sqrt{b}} \Phi(\mathbf{w}^{(i,k)})^{1/2} \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \end{aligned}$$

where the second step follows from $(1 + \delta)\bar{\mathbf{r}} \geq \mathbf{r}^{(i,k)} = \mathbf{w}^{(i,k)} + \frac{\epsilon}{n}\Phi(\mathbf{w}^{(i,k)}) \geq \mathbf{w}^{(i,k)} \geq 0$ since $\mathbf{w}^{(i,k)} \geq 0$ and hence $\Psi(\bar{\mathbf{r}}^{(i,k)}) \geq 0$.

Similarly let $\mathbf{g} = \bar{\mathbf{r}}^{(i,k)}$ we have

$$\begin{aligned} \left| \sum_e \bar{\mathbf{r}}_e^{(i,k)} \cdot (\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}) \right| &\leq \frac{C_2}{\sqrt{b}} \left(\sum_e \frac{(\bar{\mathbf{r}}_e^{(i,k)})^2}{\bar{\mathbf{r}}_e^{(i,k)}} \right)^{1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \\ &\leq \frac{C_2(1 + \epsilon)}{\sqrt{b}} \Phi(\mathbf{w}^{(i,k)})^{1/2} \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \end{aligned}$$

where the second step follows from $\sum_e \mathbf{r}_e^{(i,k)} = \Phi(\mathbf{w}^{(i,k)}) + n \cdot \frac{\epsilon}{n}\Phi(\mathbf{w}^{(i,k)})$.

Part 4 (Symmetry). It suffices to prove that for any vectors $\mathbf{g}, \mathbf{h} \in \mathbb{R}^n$, the distribution of $\mathbf{g}^\top (\mathbf{S}^\top \mathbf{S} - I) \mathbf{h}$ is symmetric around zero when $\mathbf{S} \in \mathbb{R}^{b \times n}$ is sampled from distribution Π such that each entry is $+\frac{1}{\sqrt{b}}$ with probability $1/2$ and $-\frac{1}{\sqrt{b}}$ with probability $1/2$.

Let $\mathbf{s}_1, \dots, \mathbf{s}_n \in \mathbb{R}^b$ denote the columns of \mathbf{S} . We have

$$\begin{aligned} \mathbf{g}^\top (\mathbf{S}^\top \mathbf{S} - I) \mathbf{h} &= \sum_{i=1}^n \sum_{j \neq i} \mathbf{g}_i \mathbf{h}_j \langle \mathbf{s}_i, \mathbf{s}_j \rangle \\ &= \sum_{i=1}^{n-1} \left\langle \mathbf{s}_i, \sum_{j=i+1}^n (\mathbf{g}_i \mathbf{h}_j + \mathbf{g}_j \mathbf{h}_i) \mathbf{s}_j \right\rangle. \end{aligned}$$

Define $x_i = \left\langle \mathbf{s}_i, \sum_{j=i+1}^n (\mathbf{g}_i \mathbf{h}_j + \mathbf{g}_j \mathbf{h}_i) \mathbf{s}_j \right\rangle$ for all $i \in [n-1]$. Observe that when fixing any vectors $\mathbf{s}_{i+1} = \mathbf{s}_{i+1}^*, \dots, \mathbf{s}_n = \mathbf{s}_n^*$, since \mathbf{s}_i is independent of them and each entry of \mathbf{s}_i is $+\frac{1}{\sqrt{b}}$ or $-\frac{1}{\sqrt{b}}$ with $1/2$ probability, we have

$$\begin{aligned} &\Pr \left[\left\langle \mathbf{s}_i, \sum_{j=i+1}^n (\mathbf{g}_i \mathbf{h}_j + \mathbf{g}_j \mathbf{h}_i) \mathbf{s}_j \right\rangle = z \mid \mathbf{s}_{i+1} = \mathbf{s}_{i+1}^*, \dots, \mathbf{s}_n = \mathbf{s}_n^* \right] \\ &= \Pr \left[\left\langle \mathbf{s}_i, \sum_{j=i+1}^n (\mathbf{g}_i \mathbf{h}_j + \mathbf{g}_j \mathbf{h}_i) \mathbf{s}_j \right\rangle = -z \mid \mathbf{s}_{i+1} = -\mathbf{s}_{i+1}^*, \dots, \mathbf{s}_n = -\mathbf{s}_n^* \right] \end{aligned}$$

So we can prove the claim by induction: First note that the term $x_{n-1} = \left\langle \mathbf{s}_{n-1}, (\mathbf{g}_i \mathbf{h}_j + \mathbf{g}_j \mathbf{h}_i) \mathbf{s}_n \right\rangle$ is symmetric around zero by the definition of \mathbf{S} . Suppose that for some i^* we have $\Pr[\sum_{i=i^*}^{n-1} x_i =$

$z] = \Pr[\sum_{i=i^*-1}^{n-1} x_i = -z]$, then we have for any $z \in \mathbb{R}$,

$$\begin{aligned} \Pr\left[\sum_{i=i^*-1}^{n-1} x_i = z\right] &= \sum_{t \in R} \Pr\left[\sum_{i=i^*}^{n-1} x_i = t\right] \cdot \Pr\left[x_{i^*-1} = z - t \mid \sum_{i=i^*}^{n-1} x_i = t\right] \\ &= \sum_{t \in R} \Pr\left[\sum_{i=i^*}^{n-1} x_i = -t\right] \cdot \Pr\left[x_{i^*-1} = -z + t \mid \sum_{i=i^*}^{n-1} x_i = -t\right] \\ &= \Pr\left[\sum_{i=i^*-1}^{n-1} x_i = -z\right] \end{aligned}$$

where the second step follows from $\Pr\left[\sum_{i=i^*}^{n-1} x_i = t\right] = \Pr\left[\sum_{i=i^*}^{n-1} x_i = -t\right]$ by induction hypothesis, and $\Pr\left[x_{i^*-1} = z - t \mid \sum_{i=i^*}^{n-1} x_i = t\right] = \Pr\left[x_{i^*-1} = -(z - t) \mid \sum_{i=i^*}^{n-1} x_i = -t\right]$ by our previous observation. \square

The variance and coordinate-wise bounds of the previous lemma only hold when the weights are non negative. Next we prove that we can maintain this non negativity as long as $|\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}|$ is bounded.

Lemma G.10 (Positivity of the weights). *Let k_i denote the number of width reduction steps taken by the algorithm when the i^{th} primal step is being executed. For all $i \in [0 : T]$, in the i -th primal iteration of Algorithm 3, if*

$$\mathbf{w}^{(i,k_i)} \geq 0, \quad \text{and} \quad |\hat{\mathbf{u}}^{(i,k_i)} - \mathbf{u}^{(i,k_i)}| \leq \frac{100C_2\sqrt{n}}{\sqrt{b\epsilon}},$$

then we have

$$\|\mathbf{u}^{(i,k_i)}\|_\infty \leq 2C_3^{1/3} \frac{n^{1/2-\eta}}{\epsilon^{1/3}}, \quad |\vec{\alpha}^{(i,k_i)} \hat{\mathbf{u}}^{(i,k_i)}| \leq 1/10, \quad \text{and} \quad \mathbf{w}^{(i+1,k_i)} \geq 0.$$

Proof. For simplicity, we will let k denote k_i . Observe that, when we do a primal step i.e., the condition on Line 14 is true, since for all e , $\bar{\mathbf{r}}^{(i,k)} \geq e^{-\delta} \mathbf{r}_e^{(i,k)} \geq \frac{e^{-\delta}\epsilon}{2n} \Phi(\mathbf{w}^{(i,k)}) \geq \frac{e^{-\delta}\epsilon}{2n} \Psi(\bar{\mathbf{r}}^{(i,k)})$,

$$\frac{e^{-\delta}\epsilon}{2n} \Psi(\bar{\mathbf{r}}^{(i,k)}) \|\mathbf{u}^{(i,k)}\|_3^3 \leq C_3 \rho \Psi(\bar{\mathbf{r}}^{(i,k)}).$$

Using the value of ρ , this implies that,

$$\|\mathbf{u}^{(i,k)}\|_\infty \leq \|\mathbf{u}^{(i,k)}\|_3 \leq \frac{6C_3^{1/3} n^{1/2-\eta}}{\epsilon^{1/3}}.$$

Then using our other assumption that $|\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}| \leq \frac{100C_2\sqrt{n}}{\sqrt{b\epsilon}}$, we have

$$|\hat{\mathbf{u}}_e^{(i,k)}| \leq |\mathbf{u}_e^{(i,k)}| + |\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}| \leq \frac{6C_3^{1/3} n^{1/2-\eta}}{\epsilon^{1/3}} + \frac{100C_2\sqrt{n}}{\sqrt{b\epsilon}}.$$

We also have

$$|\vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)}| \leq \alpha \cdot \left(\frac{6C_3^{1/3} n^{1/2-\eta}}{\epsilon^{1/3}} + \frac{100C_2\sqrt{n}}{\sqrt{b\epsilon}} \right) \leq 1/10.$$

If $\hat{\mathbf{u}}_e^{(i,k)} \geq 0$, then we directly have $\mathbf{w}_e^{(i+1,k)} \geq \mathbf{w}_e^{(i,k)} \geq 0$. If $\hat{\mathbf{u}}_e^{(i,k)} < 0$, then

$$\begin{aligned} \mathbf{w}_e^{(i+1,k)} &= \mathbf{w}_e^{(i,k)} \cdot (1 + \epsilon \vec{\alpha}_e^{(i,k)} \hat{\mathbf{u}}_e^{(i,k)}) \\ &\geq \mathbf{w}_e^{(i,k)} \cdot (1 - \epsilon/10) > 0. \end{aligned} \quad \square$$

Next we use the above basic properties of the approximate vectors $\hat{\mathbf{u}}^{(i,k)}$'s to prove a concentration property of the sum of the $\hat{\mathbf{u}}^{(i,k)}$'s over all iterations. Our proof crucially uses the following concentration inequality of martingales:

Lemma G.11 (Freedman's inequality, [Fre75]). *Consider a martingale Y_0, Y_1, \dots, Y_n with difference sequence X_1, X_2, \dots, X_n , i.e., $Y_0 = 0$, and for all $i \in [n]$, $Y_i = Y_{i-1} + X_i$ and $\mathbb{E}_{i-1}[Y_i] = Y_{i-1}$. Suppose $|X_i| \leq R$ almost surely for all $i \in [n]$. Define the predictable quadratic variation process of the martingale as $W_i = \sum_{j=1}^i \mathbb{E}_{j-1}[X_j^2]$, for all $i \in [n]$. Then for all $u \geq 0$, $\sigma^2 > 0$,*

$$\Pr\left[\exists i \in [n] : |Y_i| \geq u \text{ and } W_i \leq \sigma^2\right] \leq 2 \exp\left(-\frac{u^2/2}{\sigma^2 + Ru/3}\right).$$

For any $i \in [0 : T]$, let k_i denote the value of the width reduction step counter k when i is being incremented, and in the next lemma for simplicity of notations we will use the superscript (i) for the variables with superscript (i, k_i) .

Lemma G.12 (Bounds of sum of $\hat{\mathbf{u}}$ over all rounds). *Let k_i denote the number of width reduction steps taken by the algorithm when the i^{th} primal step is being executed. Let $a_0, \dots, a_T \in \mathbb{R}$ be an arbitrary sequence such that each a_i only depends on $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$ and each $|a_i| \leq C_a$. Then, for all $i \in [0 : T-1]$, the vector $\hat{\mathbf{u}}^{(i, k_i)} = (\bar{\mathbf{R}}^{(i, k_i)})^{-1/2} \cdot (\mathbf{S}^{(i)})^\top \mathbf{S}^{(i)} \cdot (\bar{\mathbf{R}}^{(i, k_i)})^{1/2} \mathbf{u}^{(i, k_i)}$ satisfies the following properties:*

$$\begin{aligned} \mathbb{E}_{\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(T-1)}} \left[\sum_{i=0}^{T-1} a_i \cdot (\hat{\mathbf{u}}_e^{(i, k_i)} - \mathbf{u}_e^{(i, k_i)}) \right] &= 0, \\ \Pr_{\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(T-1)}} \left[\left| \sum_{i=0}^{T-1} a_i \cdot (\hat{\mathbf{u}}_e^{(i, k_i)} - \mathbf{u}_e^{(i, k_i)}) \right| \leq \frac{10C_a(C_1 + C_2) \log n \cdot \sqrt{nT}}{\sqrt{b\epsilon}} \right] &\geq 1 - 1/n^3. \end{aligned}$$

Proof. Consider a fixed $e \in [n]$. Define the following truncated sequence: for $\tau^{(i)} = \hat{\mathbf{u}}_e^{(i, k_i)} - \mathbf{u}_e^{(i, k_i)}$,

$$\bar{\tau}^{(i)} = \begin{cases} \tau^{(i)} & \text{if } |\tau^{(i')}| \leq \frac{C_2 \sqrt{n}}{\sqrt{b\epsilon}} \text{ for all } i' \leq i \\ 0 & \text{otherwise.} \end{cases}$$

Also define $y^{(0)} = 0$ and $y^{(i+1)} = y^{(i)} + a_i \cdot \bar{\tau}^{(i)}$. We use the notation $\mathbb{E}_i[\cdot] = \mathbb{E}[\cdot \mid \mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i)}]$ to denote the expectation conditioned on $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i)}$.

From Part 1 of Lemma G.9 we have $\mathbb{E}_{i-1}[\tau^{(i)}] = 0$, and since a_i only depends on $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$, we also have $\mathbb{E}_{i-1}[a_i \cdot \tau^{(i)}] = 0$, and so we have $\mathbb{E}_{\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(T-1)}} [\sum_{i=0}^{T-1} a_i \cdot (\hat{\mathbf{u}}_e^{(i, k_i)} - \mathbf{u}_e^{(i, k_i)})] = 0$. From Part 4 of Lemma G.9 we have that conditioned on $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i)}$, $\Pr[\tau^{(i)} \geq \frac{C_2 \sqrt{n}}{\sqrt{b\epsilon}}] = \Pr[\tau^{(i)} \leq -\frac{C_2 \sqrt{n}}{\sqrt{b\epsilon}}]$, and therefore $\mathbb{E}_{i-1}[\bar{\tau}^{(i)}] = 0$, and $\mathbb{E}_{i-1}[a_i \cdot \bar{\tau}^{(i)}] = 0$. So we have that the sequence $y^{(0)}, \dots, y^{(T)}$ is a martingale.

Next we bound the quadratic variation. For any $i \in [T]$, if there exist $i' \leq i$ such that $|\tau^{(i')}| > \frac{C_2 \sqrt{n}}{\sqrt{b\epsilon}}$, then we have $(\bar{\tau}^{(i)})^2 = 0$. Otherwise by Lemma G.10 we have $\mathbf{w}^{(i, k_i)} \geq 0$, and since $\mathbb{E}_{i-1}[\hat{\mathbf{u}}_e^{(i, k_i)}] = \mathbf{u}_e^{(i, k_i)}$, we have

$$\mathbb{E}_{i-1}[(\tau^{(i)})^2] = \text{Var}_{\mathbf{S}^{(i)}}[\hat{\mathbf{u}}_e^{(i, k_i)} \mid \mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}] \leq \frac{C_1 n}{\epsilon b}$$

where the last step follows from Part 2 of Lemma G.9. Combining these two cases we have $\mathbb{E}_{i-1}[(\bar{\tau}^{(i)})^2] \leq \frac{C_1 n}{\epsilon b}$, and so we have $\mathbb{E}_{i-1}[(a_i \bar{\tau}^{(i)})^2] \leq C_a^2 \cdot \frac{C_1 n}{\epsilon b}$. Define $W_i = \sum_{j=1}^i \mathbb{E}_{j-1}[(a_j \bar{\tau}^{(j)})^2]$, and we have

$$W_i \leq \frac{C_a^2 C_1 n i}{\epsilon b}.$$

Using Freedman's inequality for our martingale $y^{(0)}, \dots, y^{(T)}$ with parameters $R = \frac{C_a C_2 \sqrt{n}}{\sqrt{b\epsilon}}$, $\sigma^2 = \frac{C_a^2 C_1 n T}{\epsilon b}$, and $u = \frac{10 C_a (C_1 + C_2) \log n \cdot \sqrt{n T}}{\sqrt{b\epsilon}}$, we have

$$\begin{aligned} \Pr[|y^{(T)}| \geq u] &= \Pr[|y^{(T)}| \geq u \text{ and } W_T \leq \sigma^2] \\ &\leq 2 \exp\left(-\frac{u^2/2}{\sigma^2 + Ru/3}\right) \leq 1/n^4. \end{aligned}$$

Finally, note that $y^{(T)} = \sum_{i=0}^{T-1} a_i \cdot (\hat{\mathbf{u}}_e^{(i,k_i)} - \mathbf{u}_e^{(i,k_i)})$ if we have $|\hat{\mathbf{u}}_e^{(i,k_i)} - \mathbf{u}_e^{(i,k_i)}| \leq \frac{C_2 \sqrt{n}}{\sqrt{b\epsilon}}$ for all i . By Part 3 of Lemma G.9 and union bound over all iterations, we have that this happens with probability at least $1 - 1/n^3$. Combining this with the above equation, we have

$$\Pr_{\mathbf{s}^{(0)}, \dots, \mathbf{s}^{(T-1)}} \left[\left| \sum_{i=0}^{T-1} a_i \cdot (\hat{\mathbf{u}}_e^{(i,k_i)} - \mathbf{u}_e^{(i,k_i)}) \right| \leq \frac{10 C_a (C_1 + C_2) \log n \cdot \sqrt{n T}}{\sqrt{b\epsilon}} \right] \geq 1 - 1/n^3.$$

□

G.3.2 Analysis of Algorithm 3

Next we analyze the guarantee and iteration complexity of Algorithm 3, we again first prove the change of the potentials Φ (Def. Eq. (3)) and Ψ (Def. Eq. (4)) as in the previous sections.

From the argument in the proof of Lemma G.12, we have that with probability at least $1 - 1/n^3$, we have $|\hat{\mathbf{u}}_e^{(i,k_i)} - \mathbf{u}_e^{(i,k_i)}| \leq \frac{C_2 \sqrt{n}}{\sqrt{b\epsilon}}$ for all i , and then by Lemma G.10 we have $\mathbf{w}^{(i,k_i)} \geq 0$ for all i . In this section, we assume that we are conditioned on this event, and the failure of this event corresponds to the failure of our algorithm, which happens with probability at most $1/n^3$, as stated in Theorem 3.3.

Change in Φ

Lemma G.13. *With probability at least $1 - 1/n^3$, after i primal steps, and k width-reduction steps, if $\alpha \rho^{1/3} \leq \frac{\epsilon^{1/3}}{10n^{1/3}}$, the potential Φ is bounded as follows:*

$$\begin{aligned} \Phi(\mathbf{w}^{(i,k)}) &\leq \left(\Phi(\mathbf{w}^{(0,0)}) \right) \left(1 + e^{\epsilon+\delta} \epsilon \alpha \cdot \left(1 + \frac{C_2}{\sqrt{b}} \right) + 2e^{\epsilon+2\delta} \epsilon^2 \alpha^2 \left(1 + \frac{C_2^2 \cdot n}{b\epsilon} \right) \right)^i \\ &\quad \left(1 + \epsilon e^{\epsilon+2\delta} \cdot (\tau^{-1} + \rho^{-2}) \right)^k. \end{aligned}$$

Furthermore, after every primal step, the potential can decrease by at most,

$$\Phi(\mathbf{w}^{(i+1,k)}) \geq \Phi(\mathbf{w}^{(i,k)}) \left(1 - e^{\epsilon+\delta} \epsilon \alpha \cdot \left(1 + \frac{C_2}{\sqrt{b}} \right) - 2e^{\epsilon+2\delta} \epsilon^2 \alpha^2 \left(1 + \frac{C_2^2 \cdot n}{b\epsilon} \right) \right).$$

Proof. We prove this claim by induction. Initially, $i = k = 0$, and $\Phi(\mathbf{w}^{(0,0)}) = 2n$, and thus, the claim holds trivially. Assume that the claim holds for some $i, k \geq 0$. We will use Φ as an abbreviated notation for $\Phi(\mathbf{w}^{(i,k)})$ below and \mathbf{w} to denote $\mathbf{w}^{(i,k)}$

Primal Step. Since we update the weights to be $\mathbf{w}^{(i+1,k)} = \mathbf{w}^{(i,k)} \cdot (1 + \epsilon \vec{\alpha}^{(i,k)}(C\Delta^{(i,k)} - \mathbf{d}))$, we have

$$\begin{aligned}\Phi(\mathbf{w}^{(i+1,k)}) &= \Phi(\mathbf{w}^{(i,k)}) + \epsilon \cdot \sum_e \mathbf{w}_e^{(i,k)} \cdot \vec{\alpha}_e^{(i,k)} \cdot \hat{\mathbf{u}}_e^{(i,k)} \\ &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon \alpha \cdot \sum_e \mathbf{w}_e^{(i,k)} \cdot \hat{\mathbf{u}}_e^{(i,k)} + \epsilon^2 \alpha^2 \cdot \sum_e \mathbf{w}_e^{(i,k)} \cdot (\hat{\mathbf{u}}_e^{(i,k)})^2,\end{aligned}\quad (17)$$

where the second step follows from our definition $\vec{\alpha}_e^{(i,k)} = \begin{cases} \alpha \cdot (1 + \epsilon \alpha \hat{\mathbf{u}}_e^{(i,k)}) & \text{if } \hat{\mathbf{u}}_e^{(i,k)} \geq 0 \\ \alpha / (1 - \epsilon \alpha \hat{\mathbf{u}}_e^{(i,k)}) & \text{else} \end{cases}$.

Next we bound the two terms in Eq. (17) separately. For the first term $\sum_e \mathbf{w}_e^{(i,k)} \cdot \hat{\mathbf{u}}_e^{(i,k)}$, using Cauchy-Schwarz inequality,

$$\begin{aligned}\sum_e \mathbf{w}_e^{(i,k)} \cdot \mathbf{u}_e^{(i,k)} &\leq \sqrt{\left(\sum_e \mathbf{w}_e^{(i,k)}\right) \left(\sum_e \mathbf{w}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2\right)} \\ &\leq \sqrt{e^\delta \Phi(\mathbf{w}^{(i,k)}) \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})} \\ &\leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}^{(i,k)}),\end{aligned}$$

where the third step follows from Lemma A.1. Next, from Part 3 of Lemma G.9, it holds that with probability $1 - 1/n^4$, $|\sum_e \mathbf{w}_e^{(i,k)} \cdot (\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)})| \leq \frac{C_2}{\sqrt{b}} \cdot \Phi(\mathbf{w}^{(i,k)})^{1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2}$. So we have,

$$\begin{aligned}\sum_e \mathbf{w}_e^{(i,k)} \cdot \hat{\mathbf{u}}_e^{(i,k)} &\leq \sum_e \mathbf{w}_e^{(i,k)} \cdot \mathbf{u}_e^{(i,k)} + \left| \sum_e \mathbf{w}_e^{(i,k)} \cdot (\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}) \right| \\ &\leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}^{(i,k)}) + \frac{C_2}{\sqrt{b}} \cdot \Phi(\mathbf{w}^{(i,k)})^{1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2} \\ &\leq e^{\epsilon+\delta} \cdot \left(1 + \frac{C_2}{\sqrt{b}}\right) \cdot \Phi(\mathbf{w}^{(i,k)}).\end{aligned}\quad (18)$$

In the last step we used that $\Psi(\bar{\mathbf{r}}) \leq e^\delta \Psi(\mathbf{r}) \leq e^{\epsilon+2\delta} \Phi(\mathbf{w})$.

For the second term $\sum_e \mathbf{w}_e^{(i,k)} \cdot (\hat{\mathbf{u}}_e^{(i,k)})^2$ in Eq. (17), we first bound $\sum_e \mathbf{w}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2$. We have

$$\sum_e \mathbf{w}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2 \leq e^\delta \sum_e \bar{\mathbf{r}}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2 = e^\delta \Psi(\bar{\mathbf{r}}^{(i,k)}) \leq e^{\epsilon+2\delta} \Phi(\mathbf{w}^{(i,k)}).$$

Next again using Part 3 of Lemma G.9 that with probability $1 - 1/n^4$, $|\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}| \leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}} \cdot \Phi(\mathbf{w}^{(i,k)})^{-1/2} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)})^{1/2}$, we get

$$\begin{aligned}
\sum_e \mathbf{w}_e^{(i,k)} \cdot (\hat{\mathbf{u}}_e^{(i,k)})^2 &\leq 2 \sum_e \mathbf{w}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)})^2 + 2 \sum_e \mathbf{w}_e^{(i,k)} \cdot (\mathbf{u}_e^{(i,k)} - \hat{\mathbf{u}}_e^{(i,k)})^2 \\
&\leq 2e^{\epsilon+2\delta} \Phi(\mathbf{w}^{(i,k)}) + 2 \sum_e \mathbf{w}_e^{(i,k)} \cdot \frac{C_2^2 \cdot n}{b\epsilon} \cdot \Phi(\mathbf{w}^{(i,k)})^{-1} \cdot \Psi(\bar{\mathbf{r}}^{(i,k)}) \\
&\leq 2e^{\epsilon+2\delta} \left(1 + \frac{C_2^2 \cdot n}{b\epsilon}\right) \cdot \Phi(\mathbf{w}^{(i,k)}). \tag{19}
\end{aligned}$$

Plugging Eq. (18) and (19) into Eq. (17), we have

$$\begin{aligned}
\Phi(\mathbf{w}^{(i+1,k)}) &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon\alpha \cdot e^{\epsilon+\delta} \cdot \left(1 + \frac{C_2}{\sqrt{b}}\right) \cdot \Phi(\mathbf{w}^{(i,k)}) + 2\epsilon^2\alpha^2 \cdot e^{\epsilon+2\delta} \left(1 + \frac{C_2^2 \cdot n}{b\epsilon}\right) \cdot \Phi(\mathbf{w}^{(i,k)}) \\
&= \Phi(\mathbf{w}^{(i,k)}) \cdot \left(1 + e^{\epsilon+\delta}\epsilon\alpha \cdot \left(1 + \frac{C_2}{\sqrt{b}}\right) + 2e^{\epsilon+2\delta}\epsilon^2\alpha^2 \left(1 + \frac{C_2^2 \cdot n}{b\epsilon}\right)\right).
\end{aligned}$$

The decrease of the potential follows from a similar proof.

Width Reduction Step. The proof is the same as that of Lemma G.3 since the algorithms are the same. \square

Change in Ψ

We recall the definitions of $\text{LASTWIDTH}(i, e)$, $\text{LAST}(i, e)$, $S_i \subseteq [2n]$, $\ell_{i,e}$, $\text{SIZE}(k)$, and $L \leq \frac{1}{100(\log^4 n)\epsilon\alpha\rho}$ from Section G.2. We will use these in the proof of the following lemma.

Lemma G.14 (Change in Ψ for Algorithm 3). *For any integer $c \geq 0$, after L primal steps from $(c-1)L$ to cL , if $\rho^2\tau^{-1} \geq 0.1$, the potential Ψ is bounded as follows:*

$$\Psi(\bar{\mathbf{r}}^{(cL, k_{cL})}) \geq \Psi(\bar{\mathbf{r}}^{((c-1)L, k_{(c-1)L})}) \cdot \left(1 - \tilde{O}(\epsilon\alpha\rho L)\right) \cdot \prod_{k=k_{(c-1)L}}^{k_{cL}} \left(1 + O\left(\frac{\epsilon^{4/3}\rho^{2/3} \cdot \text{SIZE}(k)^{1/3}}{n^{1/3} \cdot \log^{2/3}(\frac{n}{\epsilon\rho})}\right)\right).$$

Proof. Primal steps. For a primal step, $\mathbf{w}_e^{(i+1, k_i)} = \mathbf{w}_e^{(i, k_i)}(1 + \epsilon\vec{\alpha}_e^{(i, k_i)}\hat{\mathbf{u}}_e^{(i, k_i)})$, we have

$$\begin{aligned}
\mathbf{r}_e^{(i+1, k_i)} - \mathbf{r}_e^{(i, k_i)} &= \mathbf{w}_e^{(i+1, k_i)} - \mathbf{w}_e^{(i, k_i)} + \frac{\epsilon}{n} \cdot (\Phi(\mathbf{w}^{(i+1, k_i)}) - \Phi(\mathbf{w}^{(i, k_i)})) \\
&= \epsilon\vec{\alpha}_e^{(i, k_i)}\hat{\mathbf{u}}_e^{(i, k_i)}\mathbf{w}_e^{(i, k_i)} + \frac{\epsilon}{n} \cdot (\Phi(\mathbf{w}^{(i+1, k_i)}) - \Phi(\mathbf{w}^{(i, k_i)}))
\end{aligned}$$

Since from Lemma G.13, $\Phi(\mathbf{w}^{(i, k_i)}) \cdot (1 - \tilde{\Omega}(\epsilon\alpha)) \leq \Phi(\mathbf{w}^{(i+1, k_i)}) \leq \Phi(\mathbf{w}^{(i, k_i)}) \cdot (1 + \tilde{O}(\epsilon\alpha))$, and $\mathbf{r}_e^{(i, k_i)} \geq \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i, k_i)})$, we have

$$\epsilon\vec{\alpha}_e^{(i, k_i)}\hat{\mathbf{u}}_e^{(i, k_i)}\mathbf{w}_e^{(i, k_i)} - \tilde{\Omega}(\epsilon\alpha) \cdot \mathbf{r}_e^{(i, k_i)} \leq \mathbf{r}_e^{(i+1, k_i)} - \mathbf{r}_e^{(i, k_i)} \leq \epsilon\vec{\alpha}_e^{(i, k_i)}\hat{\mathbf{u}}_e^{(i, k_i)}\mathbf{w}_e^{(i, k_i)} + \tilde{O}(\epsilon\alpha) \cdot \mathbf{r}_e^{(i, k_i)}. \tag{20}$$

Since $|\vec{\alpha}_e^{(i, k_i)}\hat{\mathbf{u}}_e^{(i, k_i)}| \leq 1/10$ by Lemma G.10, and $\mathbf{w}_e^{(i, k_i)} \leq \mathbf{r}_e^{(i, k_i)}$ we also have

$$(1 - \epsilon)\mathbf{r}_e^{(i, k_i)} \leq \mathbf{r}_e^{(i+1, k_i)} \leq (1 + \epsilon)\mathbf{r}_e^{(i, k_i)}. \tag{21}$$

Next we consider the two cases that could happen to the coordinate e in the i -th primal step. From now on, when it's clear from the context, we will use $\mathbf{r}_e^{(i)}$ to refer to $\mathbf{r}_e^{(i, k_{i-1})}$, and similarly $\bar{\mathbf{r}}_e^{(i)}$ to refer to $\bar{\mathbf{r}}_e^{(i, k_{i-1})}$, so that this is consistent with the notations used in SELECTVECTOR.

1. If SELECTVECTOR doesn't update $\bar{\mathbf{r}}_e$ on the i -th iteration, then we have $\bar{\mathbf{r}}_e^{(i+1,k_i)} = \bar{\mathbf{r}}_e^{(i,k_i)}$.
2. If SELECTVECTOR updates $\bar{\mathbf{r}}_e$ on the i -th iteration, i.e., $e \in S_i$, then we make the same definitions as the proof of Lemma G.5: we define $j_{i,e} := \max\{\text{LASTWIDTH}(i, e), \text{LAST}(i, e)\}$, i.e., $j_{i,e} \leq i$ is the last primal iterate during which the algorithm updates \mathbf{w}_e . We also define $\ell_{i,e}$ to be the smallest integer ℓ such that $i+1 \equiv 0 \pmod{2^\ell}$ and $|\ln(\frac{\mathbf{r}_e^{(i+1)}}{\mathbf{r}_e^{(i+1-2^\ell)}})| \geq \frac{\delta}{2 \log n}$.

By definition we have $\bar{\mathbf{r}}_e^{(i+1,k_i)} = \mathbf{r}_e^{(i+1,k_i)}$, and $\bar{\mathbf{r}}_e^{(i,k_i)} = \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$.

Then by the same argument as Lemma G.5, in this case we have the following properties:

- For all $j \in [j_{i,e} + 1, i]$, the value of $\bar{\mathbf{r}}_e$ remains the same for all width reduction steps between the $(j-1)^{\text{th}}$ and j^{th} primal steps, i.e., $\bar{\mathbf{r}}_e^{(j,k_{j-1})} = \bar{\mathbf{r}}_e^{(j,k_j)}$.
- For all $j \in [j_{i,e} + 1, i]$, $\mathbf{r}_e^{(j,k_j)} \approx_\delta \bar{\mathbf{r}}_e^{(j,k_j)} = \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$.
- If $i+1-2^{\ell_{i,e}} > j_{i,e}$, then $|\mathbf{r}_e^{(i+1-2^{\ell_{i,e}})} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}| \leq 5 \log n \cdot |\mathbf{r}_e^{(i+1)} - \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}|$.

And same as Lemma G.5, we can without loss of generality assume that $j_{i,e} \geq i+1-2^{\ell_{i,e}}$, since otherwise we can upper bound $|\bar{\mathbf{r}}_e^{(i+1,k_i)} - \bar{\mathbf{r}}_e^{(i,k_i)}|$ by $\tilde{O}(|\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i+1-2^{\ell_{i,e}})}|)$ instead of $|\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}|$.

Abbreviating $\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}$ as $\mathbf{u}^{(i,k_i)}$, now we have

$$\begin{aligned}
\Psi(\bar{\mathbf{r}}^{(i+1,k_i)}) &\stackrel{(i)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \sum_e \left(\frac{\bar{\mathbf{r}}_e^{(i+1,k_i)} - \bar{\mathbf{r}}_e^{(i,k_i)}}{\bar{\mathbf{r}}_e^{(i+1,k_i)}} \right) \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\
&\stackrel{(ii)}{=} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \sum_{e \in S_i} \left(\frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}}{\mathbf{r}_e^{(i+1,k_i)}} \right) \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\
&\stackrel{(iii)}{=} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \frac{(\mathbf{r}_e^{(j+1,k_j)} - \mathbf{r}_e^{(j,k_j)})}{\mathbf{r}_e^{(i+1,k_i)}} \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\
&\stackrel{(iv)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \frac{\epsilon \vec{\alpha}_e^{(j,k_j)} \hat{\mathbf{u}}_e^{(j,k_j)} \mathbf{w}_e^{(j,k_j)} + \tilde{O}(\epsilon \alpha) \cdot \mathbf{r}_e^{(j,k_j)}}{\mathbf{r}_e^{(i+1,k_i)}} \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\
&\stackrel{(v)}{=} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \tilde{O}(\epsilon \alpha) \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\
&\quad - \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \frac{\epsilon \vec{\alpha}_e^{(j,k_j)} \hat{\mathbf{u}}_e^{(j,k_j)} \mathbf{w}_e^{(j,k_j)}}{\mathbf{r}_e^{(i+1,k_i)}} \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\
&\stackrel{(vi)}{\geq} \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - \tilde{O}(\epsilon \alpha) \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\
&\quad - \sum_{e \in S_i} \sum_{j=j_{i,e}}^i \frac{\epsilon \vec{\alpha}_e^{(j,k_j)} \mathbf{w}_e^{(j,k_j)}}{\mathbf{r}_e^{(i+1,k_i)}} \bar{\mathbf{r}}_e^{(i,k_i)} |\mathbf{u}_e^{(j,k_j)}| (\mathbf{u}_e^{(i,k_i)})^2 \\
&\quad - \epsilon \sum_{e \in S_i} \left| \sum_{j=j_{i,e}}^i \vec{\alpha}_e^{(j,k_j)} (\hat{\mathbf{u}}_e^{(j,k_j)} - \mathbf{u}_e^{(j,k_j)}) \mathbf{w}_e^{(j,k_j)} \right| \frac{\bar{\mathbf{r}}_e^{(i,k_i)}}{\mathbf{r}_e^{(i+1,k_i)}} (\mathbf{u}_e^{(i,k_i)})^2.
\end{aligned}$$

Here (i) follows from Lemma A.3, (ii) follows from $\bar{\mathbf{r}}_e^{(i+1,k_i)} = \mathbf{r}_e^{(i+1,k_i)}$, and $\bar{\mathbf{r}}_e^{(i,k_i)} = \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$ for $e \in S_i$, (iv) follows from Eq. (20). To obtain (v) we use, $\mathbf{r}_e^{(j+1,k_j)} \approx_\delta \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$, $\mathbf{r}_e^{(i,k_i)} \approx_\delta \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})}$ as we argued above, and $\mathbf{r}_e^{(i+1,k_i)} \approx_\epsilon \mathbf{r}_e^{(i,k)}$ from Eq. (21). Combining these we have $\mathbf{r}_e^{(j,k_j)} \approx_{3\epsilon} \mathbf{r}_e^{(i+1,k)}$. Further, (vi) follows from splitting $\hat{\mathbf{u}}_e^{(j,k_j)} = \mathbf{u}_e^{(j,k_j)} + (\hat{\mathbf{u}}_e^{(j,k_j)} - \mathbf{u}_e^{(j,k_j)})$.

Now, we can bound the third term using AM-GM inequality $-|\tilde{\mathbf{C}}\Delta^{(j,k_j)} - \tilde{\mathbf{d}}|_e(\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}})_e^2 \leq \frac{1}{3} \cdot |\tilde{\mathbf{C}}\Delta^{(j,k_j)} - \tilde{\mathbf{d}}|_e^3 + \frac{2}{3} \cdot |\tilde{\mathbf{C}}\Delta^{(i,k_i)} - \tilde{\mathbf{d}}|_e^3$, use that $\mathbf{r}_e^{(j,k_j)} \approx_\delta \mathbf{r}_e^{(j_{i,e},k_{j_{i,e}})} \approx_\delta \bar{\mathbf{r}}_e^{(i,k_i)}$ for all $e \in S_i$ and $j \in [j_{i,e}, i]$, and bound the fourth term by Lemma G.12,

$$\left| \sum_{j=j_{i,e}}^i \bar{\alpha}_e^{(j,k_j)} (\hat{\mathbf{u}}_e^{(j,k)} - \mathbf{u}_e^{(j,k)}) \mathbf{w}_e^{(j,k_j)} \right| \leq \tilde{O}(\alpha) \sqrt{\frac{n(i+1-j_{i,e})}{b\epsilon}} \cdot \mathbf{r}_e^{(i+1,k_i)},$$

to get,

$$\begin{aligned} \Psi(\bar{\mathbf{r}}^{(i+1,k_i)}) &\geq \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - O(\epsilon\alpha) \cdot \sum_{e \in S_i} (i+1-j_{i,e}) \cdot \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2 \\ &\quad - O(\epsilon\alpha) \cdot \sum_{e \in S_i} (i+1-j_{i,e}) \cdot \bar{\mathbf{r}}_e^{(i,k_i)} |\mathbf{u}_e^{(i,k_i)}|^3 - O(\epsilon\alpha) \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^{i-1} \bar{\mathbf{r}}_e^{(j,k_j)} |\mathbf{u}_e^{(i,k_i)}|^3 \\ &\quad - \tilde{O}(\epsilon\alpha) \cdot \sum_{e \in S_i} \sqrt{\frac{n(i+1-j_{i,e})}{b\epsilon}} \cdot \bar{\mathbf{r}}_e^{(i,k_i)} (\mathbf{u}_e^{(i,k_i)})^2, \end{aligned}$$

Similar to the proof of Lemma G.5, abusing the notation, let ℓ_i denote the largest integer such that $i+1 \equiv 0 \pmod{2^{\ell_i}}$. Since we assumed that $j_{i,e} \geq i+1-2^{\ell_{i,e}}$ for all $e \in S_i$, we have $i+1-j_{i,e} \leq 2^{\ell_{i,e}} \leq 2^{\ell_i}$. Also note that in primal steps we have $\sum_e \bar{\mathbf{r}}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \leq 2\rho\Psi(\bar{\mathbf{r}}^{(i,k)})$, and further note that $\sqrt{\frac{n2^{\ell_i}}{b}} \leq \rho \cdot 2^{\ell_i}$ since $b = \tilde{\Theta}(n^{1/2+\eta})$ and $\rho = \tilde{\Theta}(n^{1/2-3\eta})$ and $\eta \leq 1/10$, so the above equation becomes, for some $C_1 = \tilde{O}(1)$

$$\Psi(\bar{\mathbf{r}}^{(i+1,k_i)}) \geq \left(1 - C_1 \cdot \epsilon\alpha\rho \cdot 2^{\ell_i}\right) \cdot \Psi(\bar{\mathbf{r}}^{(i,k_i)}) - C_1 \cdot \epsilon\alpha \cdot \sum_{e \in S_i} \sum_{j=j_{i,e}}^{i-1} \bar{\mathbf{r}}_e^{(j,k_j)} |\tilde{\mathbf{C}}\Delta^{(j,k_j)} - \tilde{\mathbf{d}}|_e^3. \quad (22)$$

The remaining proof is the same as in the proof of Lemma G.5.

Width Reduction Step. The proof is the same as that of Lemma G.3 since the algorithms are the same. \square

Proof of Theorem 3.3

Proof. Let $\hat{\mathbf{x}} = \frac{\mathbf{x}^{(T)}}{T}$ be the solution returned by Algorithm 3. We will first prove that $\Phi(\mathbf{w}^{(T,K)}) \leq n^{\tilde{O}(1/\epsilon)}$. Then, from Lemma G.6, the number of width reduction steps are bounded by $\tilde{O}\left(\frac{n^{1/3}\rho^{1/3}}{\epsilon^{10/3}}\right) \leq \tilde{O}(\tau + \rho^2)$. We will then show how to bound the objective value at $\hat{\mathbf{x}}$.

Since Algorithm 3 has at most $\alpha^{-1} \log n / \epsilon^2$ primal steps, and suppose the algorithm has at most $\tilde{O}(\tau + \rho^2)$ width reduction steps⁶, from Lemma G.13,

$$\Phi(\mathbf{w}^{(T,K)}) \leq 2n \cdot e^{(1+\tilde{O}(\epsilon\alpha)) \cdot \alpha^{-1} \frac{\log n}{\epsilon^2} + \frac{K}{(\tau+\rho^2)}} \leq n^{\tilde{O}(\frac{1}{\epsilon})}.$$

⁶Similar to Lemma G.6, it is sufficient to argue that this is a sufficient upper bound on the number of width reduction steps

We can now follow the same argument as in Lemma G.7 up to Eq, (16) to get,

$$\left| \sum_{i=0}^{T-1} \hat{\mathbf{u}}_e^{(i)} \right| = \left| \sum_{i=0}^{T-1} (\mathbf{C} \Delta^{(i, k_i)} - \mathbf{d})_e \right| \leq \frac{\ln(\Phi(\mathbf{w}^{(T, K)}))}{\epsilon(1 - \epsilon)\alpha}.$$

Additionally, from Lemma G.12 we have that with probability $1 - 1/n^2$, for all $e \in [n]$

$$\left| \sum_{i=0}^{T-1} (\hat{\mathbf{u}}_e^{(i)} - \mathbf{u}_e^{(i)}) \right| \leq \frac{10(C_1 + C_2) \log n \cdot \sqrt{nT}}{\sqrt{b\epsilon}}.$$

Now,

$$\begin{aligned} \left| \sum_{i=0}^{T-1} \mathbf{u}_e^{(i)} \right| &\leq \left| \sum_{i=0}^{T-1} \hat{\mathbf{u}}_e^{(i)} \right| + \left| \sum_{i=0}^{T-1} (\hat{\mathbf{u}}_e^{(i)} - \mathbf{u}_e^{(i)}) \right| \\ &\leq \frac{\ln(\Phi(\mathbf{w}^{(T, K)}))}{\epsilon(1 - \epsilon)\alpha} + \frac{10(C_1 + C_2) \log n \cdot \sqrt{nT}}{\sqrt{b\epsilon}}. \end{aligned}$$

So we have for $b \geq \frac{n\alpha \log n}{\epsilon}$,

$$\begin{aligned} \|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty &= \frac{1}{T} \max_e \left| \sum_{i=0}^{T-1} \mathbf{u}_e^{(i)} \right| \\ &\leq \frac{\ln(\Phi(\mathbf{w}^{(T, K)}))}{\epsilon(1 - \epsilon)\alpha T} + \frac{10(C_1 + C_2) \log n \cdot \sqrt{n}}{\sqrt{b\epsilon T}} \\ &\leq 1 + 10\epsilon + O(\epsilon) \\ &\leq 1 + O(\epsilon). \end{aligned}$$

Therefore, the total number of iterations for $\eta = 1/10$ is,

$$T + K \leq \tilde{O}(1) \left(\alpha^{-1} \epsilon^{-2} + \frac{n^{1/3} \rho^{1/3}}{\epsilon^{10/3}} \right) = \tilde{O}(1) n^{2/5} \epsilon^{-4}.$$

Similar to the proof of Theorem G.2, we also have $|H| \leq \tilde{O}(n^{1/2+\eta})$. □

H Stability Guarantees of Algorithms 1 and 3

In this section we prove the stability guarantees of Algorithms 1 and 3.

For primal steps, we will prove that the ℓ_2 or ℓ_3 norm of the relative changes in resistances are bounded, and this means we can maintain a coordinate-wise approximation of the resistances under a low-rank update scheme.

For width reduction steps, we directly prove that the number of coordinates updated in each iteration follow the same low-rank update scheme.

In Section H.1, we prove the stability guarantees of the iterates of the multiplicative weights update algorithm with monotone weights given in Algorithm 1. Both primal and width reduction steps of this algorithm satisfy the stronger ℓ_3 -stability guarantee, i.e., the ℓ_3 norm of the relative changes in resistances is bounded.

In Section H.2, we prove that the primal steps of Algorithm 11 satisfy the weaker ℓ_2 -stability guarantee, and we also prove that its width reduction steps follow a low-rank update scheme.

In Section H.3, we prove a robust ℓ_2 -stability guarantee for the primal steps of Algorithm 3, and the width reduction steps are the same as that of Algorithm 11.

H.1 Stability Guarantees of Algorithm 1

Lemma 3.1 (Stability bound of ℓ_3 norm over all primal iterations). *Let k_i denote the number of width reduction steps taken by the algorithm when the i^{th} primal step is being executed. Then over all T primal steps of Algorithm 1, we have*

$$\sum_{i=0}^{T-1} \sum_{e \in S_i} \left(\frac{\mathbf{r}_e^{(i+1, k_i)} - \mathbf{r}_e^{(i, k_i)}}{\mathbf{r}_e^{(i, k_i)}} \right)^3 \leq \tilde{O}(\alpha^2 n) = \tilde{O}(n^{1/3} \epsilon^{2/3}).$$

Here S_i is the set of coordinates e at primal iteration i such that $\mathbf{r}_e^{(i+1, k_i)} \geq \mathbf{r}_e^{(i, k_i)}(1 + 3\epsilon\alpha)^7$.

Proof. Note that from Lemma A.3, for $k = k_i$ in each primal iteration the potential is increased by at least

$$\begin{aligned} \Psi(\bar{\mathbf{r}}^{(i+1, k)}) - \Psi(\bar{\mathbf{r}}^{(i, k)}) &\geq \sum_e \left(1 - \frac{\bar{\mathbf{r}}_e^{(i, k)}}{\bar{\mathbf{r}}_e^{(i+1, k)}} \right) \bar{\mathbf{r}}_e^{(i, k)} (C\Delta^{(i, k)} - \mathbf{d})_e^2 \\ &= \sum_e \left(\frac{\bar{\mathbf{r}}_e^{(i+1, k)} - \bar{\mathbf{r}}_e^{(i, k)}}{\bar{\mathbf{r}}_e^{(i, k)}} \right) \frac{(\bar{\mathbf{r}}_e^{(i, k)})^2}{\bar{\mathbf{r}}_e^{(i+1, k)}} (C\Delta^{(i, k)} - \mathbf{d})_e^2. \end{aligned} \quad (23)$$

We know from the algorithm that for a primal step, $\mathbf{w}^{(i+1, k)} = \mathbf{w}^{(i, k)}(1 + \epsilon\alpha|C\Delta^{(i, k)} - \mathbf{d}|)$. We will use this to compute the relative change in resistances. Let $\mathbf{r}' = \mathbf{r}^{(i+1, k)}$, $\mathbf{r} = \mathbf{r}^{(i, k)}$ and $\Delta = \Delta^{(i, k)}$.

$$\begin{aligned} \frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}_e} &= \frac{\mathbf{w}'_e - \mathbf{w}_e}{\mathbf{r}_e} + \frac{\epsilon}{n} \frac{\Phi(\mathbf{w}') - \Phi(\mathbf{w})}{\mathbf{r}_e} \\ &\leq \epsilon\alpha|C\Delta - \mathbf{d}|_e + \frac{\epsilon}{n} \frac{\Phi(\mathbf{w}') - \Phi(\mathbf{w})}{\mathbf{r}_e}. \end{aligned}$$

From the above, we note that,

$$\begin{aligned} |C\Delta - \mathbf{d}|_e^2 &\geq \left(\frac{1}{\epsilon\alpha} \frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}_e} - \frac{1}{\alpha n} \frac{\Phi(\mathbf{w}') - \Phi(\mathbf{w})}{\mathbf{r}_e} \right)^2 \\ &\geq \frac{1}{\epsilon^2 \alpha^2} \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}_e} \right)^2 - 2 \frac{1}{\epsilon \alpha^2 n} \frac{(\Phi(\mathbf{w}') - \Phi(\mathbf{w}))(\mathbf{r}'_e - \mathbf{r}_e)}{\mathbf{r}_e^2}. \end{aligned}$$

We know that for a primal step, $\Phi(\mathbf{w}') - \Phi(\mathbf{w}) \leq (1 + \epsilon)\epsilon\alpha\Phi(\mathbf{w})$. Using this and $e \in S_i$, the above becomes,

$$|C\Delta - \mathbf{d}|_e^2 \geq \frac{1}{3\epsilon^2 \alpha^2} \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}_e} \right)^2$$

⁷We note that it is sufficient to consider these sets S_i 's since any change that is smaller than the ones captured here can happen only $\tilde{O}(1)$ times.

Now, for e , let i'_e denote the last primal iterate where e was updated via a width reduction step. Now, since the \mathbf{r} 's are increasing,

$$\frac{\bar{\mathbf{r}}_e^{(i+1,k_i)} - \bar{\mathbf{r}}_e^{(i,k_i)}}{\bar{\mathbf{r}}_e^{(i,k_i)}} = \sum_{j=i'}^i \frac{\bar{\mathbf{r}}_e^{(j+1,k_j)} - \bar{\mathbf{r}}_e^{(j,k_j)}}{\bar{\mathbf{r}}_e^{(j,k_j)}} \geq \frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i,k_i)}}{\mathbf{r}_e^{(i,k_i)}}.$$

Using these bounds in Eq. (23),

$$\begin{aligned} \Psi(\bar{\mathbf{r}}^{(i+1,k)}) &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \sum_e \left(\frac{\bar{\mathbf{r}}_e^{(i+1,k)} - \bar{\mathbf{r}}_e^{(i,k)}}{\bar{\mathbf{r}}_e^{(i,k)}} \right) \frac{(\bar{\mathbf{r}}_e^{(i,k)})^2}{\bar{\mathbf{r}}_e^{(i+1,k)}} (C\Delta^{(i,k)} - \mathbf{d})_e^2 \\ &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \frac{(1-\delta)}{\epsilon^2 \alpha^2} \sum_e \left(\frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i,k_i)}}{\mathbf{r}_e^{(i,k_i)}} \right)^3 \bar{\mathbf{r}}_e^{(i,k)} \\ &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \frac{(1-\delta)^3}{\epsilon \alpha^2 n} \sum_e \left(\frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i,k_i)}}{\mathbf{r}_e^{(i,k_i)}} \right)^3 \Psi(\bar{\mathbf{r}}_e^{(i,k)}) \\ &\geq \Psi(\bar{\mathbf{r}}^{(i,k)}) \left(1 + \frac{1}{10\epsilon \alpha^2 n} \sum_e \left(\frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i,k_i)}}{\mathbf{r}_e^{(i,k_i)}} \right)^3 \right) \end{aligned}$$

We can now recurse on the above and get,

$$\Psi(\mathbf{r}^{(T,K)}) \geq \Psi(\mathbf{r}^{(0,0)}) \prod_{t \geq 0} \left(1 + \frac{1}{\epsilon \alpha^2 n} \sum_{e \in S_t} \left(\frac{\mathbf{r}_e^{(t+1,k_t)}}{\mathbf{r}_e^{(t,k_t)}} - 1 \right)^3 \right). \quad (24)$$

Taking logs,

$$\log \frac{\Psi(\mathbf{r}^{(T,K)})}{\Psi(\mathbf{r}^{(0,0)})} \geq \frac{1}{\epsilon(1+\epsilon)\alpha^2 n} \sum_{t \geq 0} \sum_{e \in S_t} \left(\frac{\mathbf{r}_e^{(t+1,k_t)}}{\mathbf{r}_e^{(t,k_t)}} - 1 \right)^3.$$

Since $\Psi(\mathbf{r}^{(T,K)}) \leq \Phi(\mathbf{w}^{(T,K)}) \leq n^{O(1/\epsilon)}$, and $\Psi(\mathbf{r}^{(0,0)}) \geq L$,

$$\sum_{t \geq 0} \sum_{e \in S_t} \left(\frac{\mathbf{r}_e^{(t+1,k_t)}}{\mathbf{r}_e^{(t,k_t)}} - 1 \right)^3 \leq \tilde{O}(\alpha^2 n).$$

□

Lemma 3.2 (Stability bound of ℓ_3 norm over all width reduction iterations). *Let i_k denote the number of primal steps taken before the execution of the k^{th} width reduction step. Then, over all K width reduction steps of Algorithm 1, we have*

$$\sum_{k=0}^{K-1} \left(\frac{\mathbf{r}_e^{(i_k,k+1)} - \mathbf{r}_e^{(i_k,k)}}{\mathbf{r}_e^{(i_k,k)}} \right)^3 \leq \tilde{O}(n^{1/3}).$$

Proof. Again from Lemma A.3,

$$\Psi(\bar{\mathbf{r}}^{(i,k+1)}) \geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \sum_e \left(\frac{\bar{\mathbf{r}}_e^{(i,k+1)} - \bar{\mathbf{r}}_e^{(i,k)}}{\bar{\mathbf{r}}_e^{(i,k)}} \right) \frac{(\bar{\mathbf{r}}_e^{(i,k)})^2}{\bar{\mathbf{r}}_e^{(i,k+1)}} (C\Delta^{(i,k)} - \mathbf{d})_e^2.$$

Now, $\bar{\mathbf{r}}_e^{(i,k+1)} \leq \bar{\mathbf{r}}_e^{(i,k)}(1+\epsilon)(1+\delta)$ and for $e \in H_k$, we know that $|\mathbf{C}\Delta^{(i,k)} - \mathbf{d}|_e \geq \tau$. Also,

$$\frac{\bar{\mathbf{r}}_e^{(i,k+1)} - \bar{\mathbf{r}}_e^{(i,k)}}{\bar{\mathbf{r}}_e^{(i,k)}} \geq \frac{\mathbf{r}_e^{(i,k+1)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}} - \frac{2\delta}{1-\delta} \geq \frac{1}{10} \frac{\mathbf{r}_e^{(i,k+1)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}}.$$

This gives us,

$$\Psi(\bar{\mathbf{r}}^{(i,k+1)}) \geq \Psi(\bar{\mathbf{r}}^{(i,k)}) + \sum_e \left(\frac{\mathbf{r}_e^{(i,k+1)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}} \right)^3 \frac{\mathbf{r}_e^{(i,k)}}{1+\epsilon} \frac{\tau^2}{\epsilon^2}.$$

Further, using that $\mathbf{r}_e^{(i,k)} \geq \epsilon \Psi(\mathbf{r}_e^{(i,k)})/n$, and $\Psi(\mathbf{r}^{(T,K)}) \leq (1+\epsilon)\Phi(\mathbf{w}^{(T,K)}) \leq n^{O(1/\epsilon)}$,

$$n^{O(1/\epsilon)} \geq L \Pi_{k \geq 0} \left(1 + \sum_e \left(\frac{\mathbf{r}_e^{(i,k+1)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}} \right)^3 \frac{1}{(1+\epsilon)\epsilon} \frac{\tau^2}{n} \right)$$

In other words,

$$\sum_k \sum_e \left(\frac{\mathbf{r}_e^{(i,k+1)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}} \right)^3 \leq \tilde{O}(n^{1/3}).$$

□

H.2 Algorithm Warm Up: Low-Rank Update Scheme

Lemma H.1 (Low-rank update scheme of width reduction steps). *Let $\eta = 1/10$. For every $\ell = 0, 1, 2, \dots, \log(\frac{10n^{2/5}}{\tau^{1/2}\epsilon^{1/2}})$, in Algorithm 11 there are at most $\frac{T}{2^\ell}$ number of width reductions steps in which $\bar{\mathbf{r}}$ receives an update of rank $O(n^{1/5}2^{2\ell} \cdot (\log n)^{28/3} \log(\frac{n}{\Psi_0})^{4/3}\epsilon^{-1})$.*

Proof. First note that from Lemma G.4, we have that for any width reduction step, the size of H satisfies $|H| \leq \frac{n}{\tau\epsilon} \cdot e^{\epsilon+2\delta}$, and hence the update to $\bar{\mathbf{r}}$ in any width reduction step has size at most $\frac{n}{\tau\epsilon} \cdot e^{\epsilon+2\delta} + 1 \leq n^{1/5}2^{2\ell_{\max}}$ where $\ell_{\max} := \log(\frac{10n^{2/5}}{\tau^{1/2}\epsilon^{1/2}})$.

Consider any fixed integer $c \in [1 : \frac{T}{L}]$. Consider any integer $\ell \in [0 : \ell_{\max}]$, and let $K_{c,\ell}$ denote the number of width steps between primal iterations $(c-1)L$ and (cL) such that $\text{SIZE}(k) \in [n^{1/5}2^{2\ell}, 4 \cdot n^{1/5}2^{2\ell}]$. Using Lemma G.5 and using a proof similar to that of Lemma G.6, we have

$$\begin{aligned} \left(1 + C_3 \frac{\epsilon^{4/3} \cdot \rho^{2/3} \cdot (n^{1/5}2^{2\ell})^{1/3}}{n^{1/3} \log^{2/3}(\frac{n}{\epsilon\rho})} \right)^{K_{c,\ell}} &\leq \frac{2\Psi(\bar{\mathbf{r}}^{(cL,k_{cL})})}{\Psi(\bar{\mathbf{r}}^{((c-1)L,k_{(c-1)L})})} \leq \frac{4\Psi(\bar{\mathbf{r}}^{(cL,k_{cL})})}{\Psi_0} \leq \frac{10n^{3\log n/\epsilon}}{\Psi_0} \\ &\Rightarrow K_{c,\ell} \leq O\left(\frac{n^{4/15} \log n}{\epsilon^{4/3} \cdot \rho^{2/3} \cdot 2^{2\ell/3}} \cdot \frac{\log n}{\epsilon} \cdot \log\left(\frac{n}{\Psi_0}\right) \right), \end{aligned}$$

where the first step follows from Lemma G.5 and that the $C_2\epsilon\alpha\rho \cdot L$ factor of Lemma G.5 is upper bounded by $1/2$ since $L \leq \frac{1}{100(\log^4 n)\epsilon\alpha\rho}$, the second step follows from the same proof as Lemma J.4 that $\Psi(\bar{\mathbf{r}}^{((c-1)L,k_{(c-1)L})}) \geq \frac{1}{1+2\epsilon} \cdot \Psi(\bar{\mathbf{r}}^{(0,0)})$ and Lemma A.2 that $\Psi(\bar{\mathbf{r}}^{(0,0)}) \geq \Psi_0$, the third step follows from $\Psi(\bar{\mathbf{r}}^{(cL,k_{cL})}) \leq e^{\epsilon+\delta}\Phi(\mathbf{w}^{(cL,k_{cL})}) \leq e^{\epsilon+\delta}n^{3\log n/\epsilon}$ by Lemma A.1 and Eq. (15) of Lemma G.6.

So over $T = \alpha^{-1}\epsilon^{-2} \log n$ primal steps, and since $L = \Theta(\frac{1}{\epsilon\alpha\rho \log^4 n})$, the total number of width reduction steps with update size in $[n^{1/5}2^{2\ell}, 4 \cdot n^{1/5}2^{2\ell}]$ is upper bounded by

$$\begin{aligned} \frac{T}{L} \cdot K_{c,\ell} &\leq T \cdot O((\log^4 n)\epsilon\alpha\rho) \cdot O\left(\frac{n^{4/15} \log n}{\epsilon^{4/3} \cdot \rho^{2/3} \cdot 2^{2\ell/3}} \cdot \frac{\log n}{\epsilon} \cdot \log\left(\frac{n}{\Psi_0}\right)\right) \\ &= T \cdot O\left(\frac{n^{4/15}\alpha\rho^{1/3}(\log n)^6}{\epsilon^{4/3} \cdot 2^{2\ell/3}} \cdot \log\left(\frac{n}{\Psi_0}\right)\right) \\ &= T \cdot O\left(\frac{n^{-1/15}(\log n)^6}{\epsilon \cdot 2^{2\ell/3}} \cdot \log\left(\frac{n}{\Psi_0}\right)\right), \end{aligned} \quad (25)$$

where the third step follows from $\alpha = O(n^{-1/2+\eta} \cdot \epsilon \cdot \log(n)^{-4/3} \log(\frac{n}{\Psi_0})^{-1/3})$ and $\rho = n^{1/2-3\eta} \cdot \epsilon^{-2} \cdot \log(n)^4 \log(\frac{n}{\Psi_0})$.

Since we only consider $\ell \leq \ell_{\max} = \log(\frac{10n^{2/5}}{\tau^{1/2}\epsilon^{1/2}})$, we have

$$2^\ell \leq O\left(\frac{n^{2/5}}{\tau^{1/2}\epsilon^{1/2}}\right) = O\left(\frac{n^{2/5}}{n^{1/4-\eta/2} \cdot \epsilon^{-3/2} \cdot \log(n)^4 \log(\frac{n}{\Psi_0})}\right) = O\left(\frac{n^{\eta/2+3/20}\epsilon^{3/2}}{\log(n)^4 \log(\frac{n}{\Psi_0})}\right),$$

where the second step follows from $\tau = n^{1/2-\eta} \cdot \epsilon^{-4} \cdot \log(n)^8 \log(\frac{n}{\Psi_0})^2$. So we have

$$O\left(\frac{n^{\eta/6+1/20}\epsilon^{1/2}}{\log(n)^{4/3} \log(\frac{n}{\Psi_0})^{1/3}}\right) \cdot \frac{1}{2^{\ell/3}} \geq 1,$$

so we can multiply this factor to the upper bound of Eq. (25), and we have that the total number of width reduction steps with update size in $[n^{1/5}2^{2\ell}, 4 \cdot n^{1/5}2^{2\ell}]$ is upper bounded by

$$\begin{aligned} &T \cdot O\left(\frac{n^{-1/15}(\log n)^6}{\epsilon \cdot 2^{2\ell/3}} \cdot \log\left(\frac{n}{\Psi_0}\right)\right) \cdot O\left(\frac{n^{\eta/6+1/20}\epsilon^{1/2}}{\log(n)^{4/3} \log(\frac{n}{\Psi_0})^{1/3}}\right) \cdot \frac{1}{2^{\ell/3}} \\ &\leq T \cdot O\left(\frac{1}{2^\ell} \cdot \frac{(\log n)^{14/3} \log(\frac{n}{\Psi_0})^{2/3}}{\epsilon^{1/2}}\right), \end{aligned}$$

where the second step follows from $\eta = 1/10$.

Letting $\ell' = \ell - \log\left(O\left(\frac{(\log n)^{14/3} \log(\frac{n}{\Psi_0})^{2/3}}{\epsilon^{1/2}}\right)\right)$, we have that there are at most $T \cdot \frac{1}{2^{\ell'}}$ width reduction steps with update size $O\left(n^{1/5}2^{2\ell'} \cdot (\log n)^{28/3} \log(\frac{n}{\Psi_0})^{4/3} \epsilon^{-1}\right)$. \square

Lemma H.2 (ℓ_2 stability of primal steps). *Algorithm 11 satisfies that for all primal steps i ,*

$$\sum_e \left(\log(\mathbf{r}_e^{(i+1,k_i)}) - \log(\mathbf{r}_e^{(i,k_i)}) \right)^2 \leq \tilde{O}(n^{2\eta}\epsilon^3).$$

Proof. We will first compute the ratio $\frac{\mathbf{r}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}}$.

$$\begin{aligned} \frac{\mathbf{r}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}} &\leq \frac{\mathbf{w}_e^{(i,k)}(1 + \epsilon \vec{\alpha}^{(i,k)} | \tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}} |_e) + \frac{\epsilon}{2n} \Phi(\mathbf{w}^{(i+1,k)})}{\mathbf{r}_e^{(i,k)}} \\ &\leq \frac{\mathbf{w}_e^{(i,k)} + \epsilon \alpha \mathbf{w}_e^{(i,k)} | \tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}} |_e + \frac{\epsilon}{2n} (1 + \epsilon \alpha e^{\epsilon+\delta}) \Phi(\mathbf{w}^{(i,k)})}{\mathbf{r}_e^{(i,k)}} \\ &\leq 1 + \epsilon \alpha | \tilde{\mathbf{C}} \Delta^{(i,k)} - \tilde{\mathbf{d}} |_e + \epsilon \alpha e^{\epsilon+\delta}, \end{aligned}$$

where the second step follows from Lemma G.3.

Now taking log and using that $\log(1+x) \leq x$ for all $x \geq -1$,

$$\log\left(\frac{\mathbf{r}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}}\right) \leq \epsilon\alpha|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e + \epsilon\alpha e^{\epsilon+\delta}.$$

Similarly we also have $\log\left(\frac{\mathbf{r}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}}\right) \geq -\epsilon\alpha|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e - \epsilon\alpha e^{\epsilon+\delta}$.

Squaring and summing over all e ,

$$\begin{aligned} \sum_e \log\left(\frac{\mathbf{r}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}}\right)^2 &\leq 2\epsilon^2\alpha^2 \sum_e |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^2 + 2\epsilon^2\alpha^2 e^{2(\epsilon+\delta)}n \\ &\leq 4\epsilon\alpha^2 n(1+\delta) \sum_e \frac{\bar{\mathbf{r}}_e^{(i,k)}|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^2}{\Phi(\mathbf{w}^{(i,k)})} + 2e^{2(\epsilon+\delta)}\epsilon^2\alpha^2 n \\ &= O(\epsilon\alpha^2 n) = O\left(n^{2\eta} \cdot \epsilon^3 \cdot \log(n)^{-8/3} \log\left(\frac{n}{\Psi_0}\right)^{-2/3}\right). \end{aligned}$$

□

From the above lemma and Lemma B.1, we directly have the following corollary.

Corollary H.3 (Low-rank update scheme of primal steps). *For every $\ell = 0, 1, \dots, \log T$, in Algorithm 11 there are at most $\frac{T}{2^\ell}$ number of primal steps in which $\bar{\mathbf{r}}$ receives an update of rank $O\left(n^{2\eta} 2^{2\ell} \cdot \epsilon \cdot \log(n)^{-2/3} \log\left(\frac{n}{\Psi_0}\right)^{-2/3}\right)$.*

H.3 Algorithm 3

Low-rank update scheme First note that the width reduction steps of Algorithm 3 are the same as Algorithm 11, so they follow the same low-rank update scheme as Lemma H.1.

Next we prove the robust ℓ_2 stability guarantees of the primal steps of Algorithm 3, and this combined with Lemma B.2 will give us the desired low-rank update scheme.

Lemma H.4 (Robust ℓ_2 stability of primal steps). *For every primal step (i, k) of Algorithm 3, define a “fake” weight:*

$$\tilde{\mathbf{r}}^{(i+1,k)} = \mathbf{r}^{(i+1,k)} - \mathbf{w}^{(i,k)}\epsilon\alpha \cdot (\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}) \cdot \frac{(1 + \vec{\alpha}^{(i,k)}\hat{\mathbf{u}}^{(i,k)})}{(1 + \alpha\hat{\mathbf{u}}^{(i,k)})}. \quad (26)$$

Every primal step of Algorithm 3 satisfies the following robust ℓ_2 stability property if $b \geq \frac{n\alpha \log^4 n}{\epsilon^3}$.

1.

$$\sum_e \ln\left(\frac{\tilde{\mathbf{r}}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}}\right)^2 \leq O(n^{2\eta}).$$

2. $\forall t \in [T], \forall e$, with probability $1 - 1/n^4$,

$$\left| \sum_{i=t'-t}^{t'} \ln\left(\frac{\tilde{\mathbf{r}}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}}\right) \right| \leq O(\epsilon).$$

Proof. Part 1 (primal steps: ℓ_2 norm of tilde version). We first note that by the definition of $\tilde{\mathbf{r}}^{(i+1,k)}$ we have

$$\begin{aligned}
& \tilde{\mathbf{r}}^{(i+1,k)} - \mathbf{r}^{(i,k)} \\
&= \mathbf{r}^{(i+1,k)} - \mathbf{r}^{(i,k)} - \epsilon\alpha \cdot \mathbf{w}^{(i,k)} \cdot (\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}) \cdot \frac{(1 + \vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)})}{(1 + \alpha \hat{\mathbf{u}}^{(i,k)})} \\
&= \mathbf{w}^{(i,k)} \cdot \epsilon \vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)} + \frac{\epsilon}{n} \left(\Phi(\mathbf{w}^{(i+1,k)}) - \Phi(\mathbf{w}^{(i,k)}) \right) \\
&\quad - \epsilon\alpha \cdot \mathbf{w}^{(i,k)} \cdot (\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}) \cdot \frac{(1 + \vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)})}{(1 + \alpha \hat{\mathbf{u}}^{(i,k)})} \\
&= \epsilon \vec{\alpha}^{(i,k)} \cdot \mathbf{w}^{(i,k)} \cdot \mathbf{u}^{(i,k)} + \epsilon (\vec{\alpha}^{(i,k)} - \alpha) \cdot \mathbf{w}^{(i,k)} \cdot (\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}) \\
&\quad + \left(1 - \frac{(1 + \vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)})}{(1 + \alpha \hat{\mathbf{u}}^{(i,k)})} \right) \cdot \epsilon\alpha \cdot \mathbf{w}^{(i,k)} \cdot (\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}) + \frac{\epsilon}{n} \left(\Phi(\mathbf{w}^{(i+1,k)}) - \Phi(\mathbf{w}^{(i,k)}) \right)
\end{aligned}$$

where we used $\mathbf{r}^{(i,k)} = \mathbf{w}^{(i,k)} + \frac{\epsilon}{m} \Phi(\mathbf{w}^{(i,k)})$ and $\mathbf{w}^{(i+1,k)} = \mathbf{w}^{(i,k)} (1 + \epsilon \vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)})$ in the second step.

Next we provide upper bounds for the terms appearing in the previous equation. By Part 3 of Lemma G.9, we have that with probability $1 - 1/n^3$, for all e we have $|\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}| \leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}}$. Then using Lemma G.10 and the definition of $\vec{\alpha}^{(i,k)}$ we have $\vec{\alpha}^{(i,k)} \leq (1 + \epsilon)\alpha$ and $|\vec{\alpha}^{(i,k)} \hat{\mathbf{u}}^{(i,k)}| \leq 0.1$. By Lemma G.13 we have with probability at least $1 - 1/n^3$, $\Phi(\mathbf{w}^{(i+1,k)}) \leq \Phi(\mathbf{w}^{(i,k)}) \cdot (1 + 10\epsilon\alpha)$. By the definition that $\vec{\alpha}_e^{(i,k)} = \begin{cases} \alpha \cdot (1 + \epsilon\alpha \hat{\mathbf{u}}_e^{(i,k)}) & \text{if } \hat{\mathbf{u}}_e^{(i,k)} \geq 0 \\ \alpha / (1 - \epsilon\alpha \hat{\mathbf{u}}_e^{(i,k)}) & \text{else} \end{cases}$, we have

$$\begin{aligned}
|\vec{\alpha}_e^{(i,k)} - \alpha| &= \begin{cases} \epsilon\alpha^2 \hat{\mathbf{u}}_e^{(i,k)} & \text{if } \hat{\mathbf{u}}_e^{(i,k)} \geq 0 \\ \epsilon\alpha^2 |\hat{\mathbf{u}}_e^{(i,k)}| \cdot (1 - \epsilon\alpha \hat{\mathbf{u}}_e^{(i,k)})^{-1} & \text{else} \end{cases} \\
&\leq (1 + \epsilon)\epsilon\alpha^2 |\hat{\mathbf{u}}_e^{(i,k)}|.
\end{aligned}$$

Plugging these upper bounds into the previous equation, we have that with probability $1 - 1/n^3$,

$$\begin{aligned}
& |\tilde{\mathbf{r}}^{(i+1,k)} - \mathbf{r}^{(i,k)}| \\
&\leq (1 + \epsilon)\epsilon\alpha \cdot \mathbf{w}^{(i,k)} \cdot |\mathbf{u}^{(i,k)}| + (1 + \epsilon)\epsilon^2\alpha^2 \cdot \mathbf{w}^{(i,k)} \cdot |\hat{\mathbf{u}}^{(i,k)}| \cdot |\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}| \\
&\quad + 2\epsilon\alpha^2 |\hat{\mathbf{u}}^{(i,k)}|^2 \cdot \epsilon\alpha \cdot \mathbf{w}^{(i,k)} \cdot |\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}| + 10\epsilon\alpha \cdot \frac{\epsilon}{n} \cdot \Phi(\mathbf{w}^{(i,k)}) \\
&\leq (1 + \epsilon)\epsilon\alpha \cdot \mathbf{w}^{(i,k)} \cdot |\mathbf{u}^{(i,k)}| + 2\epsilon^2\alpha^2 \cdot \mathbf{w}^{(i,k)} \cdot |\hat{\mathbf{u}}^{(i,k)}| \cdot |\hat{\mathbf{u}}^{(i,k)} - \mathbf{u}^{(i,k)}| + 10\epsilon\alpha \cdot \frac{\epsilon}{n} \cdot \Phi(\mathbf{w}^{(i,k)}) \\
&\leq (1 + \epsilon)\epsilon\alpha \left(1 + \frac{2C_2\sqrt{n\epsilon\alpha}}{\sqrt{b}} \right) \cdot \mathbf{w}^{(i,k)} \cdot |\mathbf{u}^{(i,k)}| + (1 + \epsilon)\epsilon\alpha^2 \cdot \mathbf{w}^{(i,k)} \cdot \frac{C_2^2 n}{b} + 10\epsilon\alpha \cdot \frac{\epsilon}{n} \cdot \Phi(\mathbf{w}^{(i,k)}).
\end{aligned}$$

Since $\mathbf{r}^{(i,k)} = \mathbf{w}^{(i,k)} + \frac{\epsilon}{n} \Phi(\mathbf{w}^{(i,k)})$,

$$\begin{aligned}
\left| \frac{\tilde{\mathbf{r}}^{(i+1,k)} - \mathbf{r}^{(i,k)}}{\mathbf{r}^{(i,k)}} \right| &\leq (1 + \epsilon)\epsilon\alpha \left(1 + \frac{2C_2\sqrt{n\epsilon\alpha}}{\sqrt{b}} \right) \cdot |\mathbf{u}^{(i,k)}| + (1 + \epsilon)\epsilon\alpha^2 \cdot \frac{C_2^2 n}{b} + 10\epsilon\alpha \\
&\leq 2\epsilon\alpha \cdot |\mathbf{u}^{(i,k)}| + O(\epsilon\alpha),
\end{aligned}$$

where the second step follows from $b \geq \frac{n\alpha \log n}{\epsilon}$.

Now taking \ln and using that $|\ln(1+x)| \leq 2|x|$ for all $|x| \geq -0.5$, we have

$$\left| \ln \left(\frac{\tilde{\mathbf{r}}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}} \right) \right| \leq 2 \left| \frac{\tilde{\mathbf{r}}_e^{(i+1,k)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}} \right| \leq 4\epsilon\alpha \cdot |\mathbf{u}_e^{(i,k)}| + O(\epsilon\alpha).$$

Squaring and summing over all e , we have

$$\begin{aligned} \sum_e \ln \left(\frac{\tilde{\mathbf{r}}_e^{(i+1,k)}}{\mathbf{r}_e^{(i,k)}} \right)^2 &\leq 32\epsilon^2\alpha^2 \sum_e (\mathbf{u}_e^{(i,k)})^2 + O(\epsilon^2\alpha^2 n) \\ &\leq 32\epsilon\alpha^2 n \sum_e \frac{\mathbf{r}_e^{(i,k)} (\mathbf{u}_e^{(i,k)})^2}{\Psi(\mathbf{r}_e^{(i,k)})} + O(\epsilon^2\alpha^2 n) \\ &\leq O(\epsilon\alpha^2 n) = O(n^{2\eta}), \end{aligned}$$

where the second step follows from $\mathbf{r}_e^{(i,k)} = \mathbf{w}_e^{(i,k)} + \frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)}) \geq \frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)})$, and the last step follows from $\alpha = \tilde{\Theta}(n^{-1/2+\eta}\epsilon)$.

Part 2 (primal steps: error of tilde version over all iterations). Consider any fixed coordinate e , and consider the t iterations between $t' - t$ to t' . We have

$$\begin{aligned} &\sum_{i=t'-t}^{t'} \ln \left(\frac{\tilde{\mathbf{r}}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}} \right) \\ &= \sum_{i=t'-t}^{t'} \ln \left(1 - \frac{\mathbf{w}_e^{(i-1,k)} \cdot \epsilon\alpha \cdot (\hat{\mathbf{u}}_e^{(i-1,k)} - \mathbf{u}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)}} \cdot \frac{(1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \hat{\mathbf{u}}_e^{(i-1,k)})} \right) \\ &\leq \epsilon\alpha \cdot \sum_{i=t'-t}^{t'} \frac{\mathbf{w}_e^{(i-1,k)} \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)}} \cdot \frac{(1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \hat{\mathbf{u}}_e^{(i-1,k)})} \\ &= \epsilon\alpha \cdot \sum_{i=t'-t}^{t'} \frac{(\mathbf{r}_e^{(i,k)} - \frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)})) \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \hat{\mathbf{u}}_e^{(i-1,k)})} \\ &\leq \epsilon\alpha \cdot \sum_{i=t'-t}^{t'} \frac{(\mathbf{r}_e^{(i,k)} - \frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)})) \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \\ &\quad + 2\epsilon\alpha^2 \cdot \sum_{i=t'-t}^{t'} \frac{|\mathbf{r}_e^{(i,k)} - \frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)})|}{\mathbf{r}_e^{(i,k)}} \cdot |\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)}|^2 \\ &\leq \epsilon\alpha \cdot \left| \sum_{i=t'-t}^{t'} \frac{\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)}}{1 + \alpha \mathbf{u}_e^{(i-1,k)}} \right| + \epsilon\alpha \cdot \left| \sum_{i=t'-t}^{t'} \frac{\frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ &\quad + 2\epsilon\alpha^2 \sum_{i=t'-t}^{t'} |\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)}|^2, \end{aligned} \tag{27}$$

where the first step follows from the definition (26) that $\tilde{\mathbf{r}}_e^{(i,k)} = \mathbf{r}_e^{(i,k)} - \mathbf{w}_e^{(i-1,k)}\epsilon\alpha \cdot (\hat{\mathbf{u}}_e^{(i-1,k)} - \mathbf{u}_e^{(i-1,k)}) \cdot \frac{(1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \hat{\mathbf{u}}_e^{(i-1,k)})}$, the second step follows from $\ln(1+x) \leq x$ for all x , the third step follows from $\mathbf{w}_e^{(i,k)} = \mathbf{w}_e^{(i-1,k)} \cdot (1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)})$ and $\mathbf{r}_e^{(i,k)} = \mathbf{w}_e^{(i,k)} + \frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)})$ and so $\mathbf{w}_e^{(i-1,k)} = \frac{\mathbf{r}_e^{(i,k)} - \frac{\epsilon}{n}\Phi(\mathbf{w}_e^{(i,k)})}{1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)}}$, the fourth step follows from $|\alpha \hat{\mathbf{u}}_e^{(i-1,k)}| \leq 0.1$ with probability

$1 - 1/n^4$ and $|\alpha \mathbf{u}_e^{(i-1,k)}| \leq 0.1$, and hence

$$\begin{aligned} & \left| \frac{1}{1 + \alpha \mathbf{u}_e^{(i-1,k)}} - \frac{1}{1 + \alpha \mathbf{u}_e^{(i-1,k)} + \alpha(\widehat{\mathbf{u}}_e^{(i-1,k)} - \mathbf{u}_e^{(i-1,k)})} \right| \\ &= \left| \frac{\alpha(\widehat{\mathbf{u}}_e^{(i-1,k)} - \mathbf{u}_e^{(i-1,k)})}{\left(1 + \alpha \mathbf{u}_e^{(i-1,k)}\right) \cdot \left(1 + \alpha \mathbf{u}_e^{(i-1,k)} + \alpha(\widehat{\mathbf{u}}_e^{(i-1,k)} - \mathbf{u}_e^{(i-1,k)})\right)} \right| \leq 2\alpha |\widehat{\mathbf{u}}_e^{(i-1,k)} - \mathbf{u}_e^{(i-1,k)}|. \end{aligned}$$

Next we bound the three terms in Eq. (27) one by one.

For the first term, since $|\alpha \mathbf{u}_e^{(i,k)}| \leq 0.1$ and each $\mathbf{u}_e^{(i,k)}$ only depends on the randomness of $\mathbf{S}^{(0)}, \dots, \mathbf{S}^{(i-1)}$, using Lemma G.12 we have

$$\epsilon \alpha \cdot \left| \sum_{i=t'-t}^{t'} \frac{\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)}}{1 + \alpha \mathbf{u}_e^{(i-1,k)}} \right| \leq \epsilon \alpha \cdot O\left(\frac{\sqrt{nt}}{\sqrt{b\epsilon}}\right) = O\left(\frac{\sqrt{nt\epsilon}\alpha}{\sqrt{b}}\right). \quad (28)$$

For the third term, since with probability $1 - 1/n^3$ we have $|\widehat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}| \leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}}$ for all i , we have

$$2\epsilon \alpha^2 \sum_{i=t'-t}^{t'} |\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)}|^2 \leq \epsilon \alpha^2 \cdot O\left(\frac{nt}{b\epsilon}\right) = O\left(\frac{nt\alpha^2}{b}\right). \quad (29)$$

It's more complicated to upper bound the second term, and we start by upper bounding the following term:

$$\begin{aligned} & \left| \sum_{i=t'-t}^{t'} \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \leq \left| \sum_{i=t'-t}^{t'} \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i-1,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \quad + \left| \sum_{i=t'-t}^{t'} \left(\frac{1}{\mathbf{r}_e^{(i,k)}} - \frac{1}{\mathbf{r}_e^{(i-1,k)}} \right) \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \leq O\left(\frac{\sqrt{nt}}{\sqrt{b\epsilon}}\right) + \left| \sum_{i=t'-t}^{t'} \frac{\overrightarrow{\alpha}_e^{(i-1,k)} \widehat{\mathbf{u}}_e^{(i-1,k)} \mathbf{w}_e^{(i-1,k)}}{\mathbf{r}_e^{(i-1,k)}} \cdot \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \quad + \left| \sum_{i=t'-t}^{t'} \frac{\frac{\epsilon}{n} (\Phi(\mathbf{w}^{(i,k)}) - \Phi(\mathbf{w}^{(i-1,k)})) \cdot \frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} \mathbf{r}_e^{(i-1,k)}} \cdot \frac{(\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \leq O\left(\frac{\sqrt{nt}}{\sqrt{b\epsilon}}\right) + \frac{\alpha t \sqrt{\epsilon n}}{\sqrt{b}} + \left| \sum_{i=t'-t}^{t'} \frac{\overrightarrow{\alpha}_e^{(i-1,k)} \widehat{\mathbf{u}}_e^{(i-1,k)} \mathbf{w}_e^{(i-1,k)}}{\mathbf{r}_e^{(i-1,k)}} \cdot \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \widehat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right|, \quad (30) \end{aligned}$$

where in the second step we upper bound the first term using Lemma G.12 since $\Phi(\mathbf{w}^{(i-1,k)})$, $\mathbf{r}_e^{(i-1,k)}$, $1 + \alpha \mathbf{u}_e^{(i-1,k)}$ are all independent of $\widehat{\mathbf{u}}_e^{(i-1,k)}$, and we upper bound the second term by

$\mathbf{r}_e^{(i,k)} = \mathbf{r}_e^{(i-1,k)} + \frac{\epsilon}{\alpha_e} \hat{\mathbf{u}}_e^{(i-1,k)} \mathbf{w}_e^{(i-1,k)} + \frac{\epsilon}{n} (\Phi(\mathbf{w}^{(i,k)}) - \Phi(\mathbf{w}^{(i-1,k)}))$ and hence

$$\frac{1}{\mathbf{r}_e^{(i,k)}} - \frac{1}{\mathbf{r}_e^{(i-1,k)}} = \frac{\frac{\epsilon}{\alpha_e} \hat{\mathbf{u}}_e^{(i-1,k)} \mathbf{w}_e^{(i-1,k)} + \frac{\epsilon}{n} (\Phi(\mathbf{w}^{(i,k)}) - \Phi(\mathbf{w}^{(i-1,k)}))}{\mathbf{r}_e^{(i,k)} \mathbf{r}_e^{(i-1,k)}},$$

and the third step follows from $\Phi(\mathbf{w}^{(i,k)}) \leq \Phi(\mathbf{w}^{(i-1,k)}) \cdot (1 + O(\epsilon\alpha))$, and $\mathbf{r}_e^{(i,k)} \geq 0.9\mathbf{r}_e^{(i-1,k)} \geq 0.9\frac{\epsilon}{n}\Phi(\mathbf{w}^{(i-1,k)})$, and with probability $1 - 1/n^3$ we have $|\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}| \leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}}$ for all i .

Note that the term $\frac{\frac{\epsilon}{\alpha_e} \hat{\mathbf{u}}_e^{(i-1,k)} \mathbf{w}_e^{(i-1,k)}}{\mathbf{r}_e^{(i-1,k)}}$ is independent of $\hat{\mathbf{u}}_e^{(i-1,k)}$ and it's at most 0.1 with probability $1 - 1/n^3$. So we can continue the same upper bound of Eq. (30) for $2 \log n$ times and we have

$$\begin{aligned} & \left| \sum_{i=t'-t}^{t'} \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \leq O\left(\left(\frac{\sqrt{nt}}{\sqrt{b\epsilon}} + \frac{\alpha t \sqrt{\epsilon n}}{\sqrt{b}}\right) \cdot \log n\right) \\ & \quad + \left| \sum_{i=t'-t}^{t'} \left(\frac{\frac{\epsilon}{\alpha_e} \hat{\mathbf{u}}_e^{(i-1,k)} \mathbf{w}_e^{(i-1,k)}}{\mathbf{r}_e^{(i-1,k)}} \right)^{2 \log n} \cdot \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \leq O\left(\left(\frac{\sqrt{nt}}{\sqrt{b\epsilon}} + \frac{\alpha t \sqrt{\epsilon n}}{\sqrt{b}}\right) \cdot \log n\right). \end{aligned} \tag{31}$$

We are finally ready to upper bound the second term of Eq. (27). We have

$$\begin{aligned} & \left| \sum_{i=t'-t}^{t'} \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i,k)}) \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)} (1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \leq \left| \sum_{i=t'-t}^{t'} \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)})}{\mathbf{r}_e^{(i,k)}} \cdot \frac{(\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \quad + \left| \sum_{i=t'-t}^{t'} \left(\frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i,k)})}{\mathbf{r}_e^{(i,k)}} - \frac{\frac{\epsilon}{n} \Phi(\mathbf{w}^{(i-1,k)})}{\mathbf{r}_e^{(i,k)}} \right) \cdot \frac{(\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \mathbf{u}_e^{(i-1,k)})} \right| \\ & \leq O\left(\left(\frac{\sqrt{nt}}{\sqrt{b\epsilon}} + \frac{\alpha t \sqrt{\epsilon n}}{\sqrt{b}}\right) \cdot \log n\right), \end{aligned} \tag{32}$$

where in the second step we upper bound the first term by Eq. (31), and we upper bound the second term by Lemma G.13 that $\Phi(\mathbf{w}^{(i,k)}) \leq \Phi(\mathbf{w}^{(i-1,k)}) \cdot (1 + O(\epsilon\alpha))$ and $\mathbf{r}_e^{(i,k)} \geq 0.9\mathbf{r}_e^{(i-1,k)} \geq 0.9\frac{\epsilon}{n}\Phi(\mathbf{w}^{(i-1,k)})$, and with probability $1 - 1/n^3$ we have $|\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}| \leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}}$ for all i .

Plugging the three upper bounds Eq. (28), (32), and (29) into Eq. (27), we have

$$\sum_{i=t'-t}^{t'} \ln\left(\frac{\tilde{\mathbf{r}}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}}\right) \leq O\left(\frac{\sqrt{nt\epsilon\alpha}}{\sqrt{b}}\right) + \epsilon\alpha \cdot O\left(\left(\frac{\sqrt{nt}}{\sqrt{b\epsilon}} + \frac{\alpha t \sqrt{\epsilon n}}{\sqrt{b}}\right) \cdot \log n\right) + O\left(\frac{nt\alpha^2}{b}\right) \leq O(\epsilon),$$

where the last step follows from $b = \frac{n\alpha \log^4 n}{\epsilon^3}$, and $t \leq T = \alpha^{-1}\epsilon^{-2} \ln n$.

We can similarly prove an upper bound of $-\sum_{i=t'-t}^{t'} \ln(\frac{\tilde{\mathbf{r}}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}})$. Similar to Eq. (27), we have

$$\begin{aligned}
-\sum_{i=t'-t}^{t'} \ln(\frac{\tilde{\mathbf{r}}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}}) &= -\sum_{i=t'-t}^{t'} \ln\left(1 - \frac{\mathbf{w}_e^{(i-1,k)} \cdot \epsilon\alpha \cdot (\hat{\mathbf{u}}_e^{(i-1,k)} - \mathbf{u}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)}} \cdot \frac{(1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \hat{\mathbf{u}}_e^{(i-1,k)})}\right) \\
&\leq \epsilon\alpha \cdot \left| \sum_{i=t'-t}^{t'} \frac{\mathbf{w}_e^{(i-1,k)} \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)}} \cdot \frac{(1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \hat{\mathbf{u}}_e^{(i-1,k)})} \right| \\
&\quad + \epsilon^2 \alpha^2 \cdot \sum_{i=t'-t}^{t'} \left(\frac{\mathbf{w}_e^{(i-1,k)} \cdot (\mathbf{u}_e^{(i-1,k)} - \hat{\mathbf{u}}_e^{(i-1,k)})}{\mathbf{r}_e^{(i,k)}} \cdot \frac{(1 + \vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)})}{(1 + \alpha \hat{\mathbf{u}}_e^{(i-1,k)})} \right)^2 \\
&\leq O(\epsilon) + O(\frac{\epsilon n t \alpha^2}{b}) \leq O(\epsilon),
\end{aligned}$$

where the second step follows from $\ln(1+x) \geq x - x^2$ for all $|x| \leq 0.5$, and in the third step we bound the first term in the same way as Eq. (27) and (28), (32), (29), and we bound the second term by the property that with probability $1 - 1/n^3$ we have $|\hat{\mathbf{u}}_e^{(i,k)} - \mathbf{u}_e^{(i,k)}| \leq \frac{C_2}{\sqrt{b}} \cdot \frac{\sqrt{n}}{\sqrt{\epsilon}}$ for all i , and $\vec{\alpha}_e^{(i-1,k)} \hat{\mathbf{u}}_e^{(i-1,k)} \leq 0.1$, $\alpha \hat{\mathbf{u}}_e^{(i-1,k)} \leq 0.1$, and $\mathbf{r}_e^{(i,k)} \geq 0.9 \mathbf{r}_e^{(i-1,k)} \geq 0.9 \mathbf{w}_e^{(i-1,k)}$. \square

Combining the above lemma and Lemma B.2, we directly have the following corollary.

Corollary H.5 (Low-rank update scheme of primal steps). *For every $\ell = 0, 1, \dots, \log T$, in Algorithm 3 there are at most $\frac{T}{2^\ell}$ number of primal steps in which $\bar{\mathbf{r}}$ receives an update of rank $O((\frac{\log n}{\delta})^2 \cdot n^{2\eta} \cdot 2^{2\ell})$.*

I Missing Proofs

Proof of Lemma A.1

Proof. We first prove that $\Psi(\bar{\mathbf{r}}) \approx_\delta \Psi(\mathbf{r})$. Let $\Delta^* := \arg \min_{\Delta \in \mathbb{R}^d} \sum_e \bar{\mathbf{r}}_e (\mathbf{C}\Delta - \mathbf{d})_e^2$. We have

$$\Psi(\mathbf{r}) = \min_{\Delta \in \mathbb{R}^d} \sum_e \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2 \leq \sum_e \mathbf{r}_e (\mathbf{C}\Delta^* - \mathbf{d})_e^2 \leq e^\delta \cdot \sum_e \bar{\mathbf{r}}_e (\mathbf{C}\Delta^* - \mathbf{d})_e^2 = e^\delta \cdot \Psi(\bar{\mathbf{r}}).$$

Similarly, we can also show $\Psi(\bar{\mathbf{r}}) \leq e^\delta \cdot \Psi(\mathbf{r})$, and so $\Psi(\bar{\mathbf{r}}) \approx_\delta \Psi(\mathbf{r})$.

Next we prove $\Psi(\bar{\mathbf{r}}) \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w})$. The following inequalities follow from Hölder's inequality and the optimum \mathbf{x}^* of the problem satisfies $\|\mathbf{C}\mathbf{x}^* - \mathbf{d}\|_\infty \leq 1$:

$$\begin{aligned}
\Psi(\bar{\mathbf{r}}) &= \min_{\Delta \in \mathbb{R}^d} \sum_{e=1}^n \bar{\mathbf{r}}_e (\mathbf{C}\Delta - \mathbf{d})_e^2 \\
&\leq \sum_e \bar{\mathbf{r}}_e (\mathbf{C}\mathbf{x}^* - \mathbf{d})_e^2 \leq e^\delta \cdot \|\mathbf{C}\mathbf{x}^* - \mathbf{d}\|_\infty^2 (\|\mathbf{w}\|_1 + \epsilon \|\mathbf{w}\|_1) \leq e^{\epsilon+\delta} \cdot \Phi(\mathbf{w}). \quad \square
\end{aligned}$$

Proof of Lemma A.2

Proof. Note that $\mathbf{r}_e^{(0,0)} = 1 + \epsilon$ for all e . Therefore,

$$\Psi(\mathbf{r}^{(0,0)}) = (1 + \epsilon) \min_{\Delta} \|\mathbf{C}\Delta - \mathbf{d}\|_2^2.$$

Using KKT conditions and the solution of the above problem that $\Delta^* = (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}^\top \mathbf{d}$, we get the required lower bound. \square

Proof of Lemma A.3

This is similar to the proof of Lemma 5.11 of [Adi+19].

Proof. We have defined,

$$\Psi(\mathbf{r}) = \min_{\Delta} \sum_e \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2.$$

This is the same as,

$$\Psi(\mathbf{r}) = \min_{\rho, \Delta} \sum_e \mathbf{r}_e \rho_e^2, \text{ s.t. } \rho = \mathbf{C}\Delta - \mathbf{d}.$$

From Lagrange duality and using strong duality,

$$\begin{aligned} \Psi(\mathbf{r}) &= \min_{\rho, \Delta} \max_{\mathbf{y}} \sum_e \mathbf{r}_e \rho_e^2 + 2\mathbf{y}^\top (\rho - \mathbf{C}\Delta + \mathbf{d}) \\ &= \max_{\mathbf{y}} \min_{\rho, \Delta} \sum_e \mathbf{r}_e \rho_e^2 + 2\mathbf{y}^\top (\rho - \mathbf{C}\Delta + \mathbf{d}). \end{aligned}$$

Using optimality conditions for Δ and ρ , we get that the optimizers ρ^* and Δ^* satisfy,

$$\mathbf{R}\rho^* = -\mathbf{y}, \quad \mathbf{C}^\top \mathbf{y} = 0.$$

Using these values back in our program, we get,

$$\Psi(\mathbf{r}) = \max_{\mathbf{y}: \mathbf{C}^\top \mathbf{y} = 0} 2\mathbf{y}^\top \mathbf{d} - \mathbf{y}^\top \mathbf{R}^{-1} \mathbf{y}. \quad (33)$$

We now note that, the following program has minimizer $(\mathbf{z}^*, \theta^*) = (\mathbf{y}^*, 1)$ and achieves the same optimum value as Program 33.

$$\Psi(\mathbf{r}) = \max_{\mathbf{z}, \theta: \mathbf{C}^\top \mathbf{z} = 0} 2\theta \mathbf{z}^\top \mathbf{d} - \theta^2 \mathbf{z}^\top \mathbf{R}^{-1} \mathbf{z}. \quad (34)$$

Since $\mathbf{z}^* = \mathbf{y}^*$ and $\theta^* = 1$ is an optimum for (34),

$$\left[\frac{d}{d\theta} \left(2\theta \mathbf{y}^{*\top} \mathbf{d} - \theta^2 \mathbf{y}^{*\top} \mathbf{R}^{-1} \mathbf{y}^* \right) \right]_{\theta=1} = 0.$$

As a result, we get that,

$$\mathbf{d}^\top \mathbf{y}^* = \mathbf{y}^{*\top} \mathbf{R}^{-1} \mathbf{y}^*,$$

and $\Psi(\mathbf{r}) = \mathbf{d}^\top \mathbf{y}^*$. Using this, we claim that the following program has objective value $2 - 1/\Psi(\mathbf{r})$:

$$2 - \frac{1}{\Psi(\mathbf{r})} = \max_{\substack{\mathbf{C}^\top \mathbf{y} = 0 \\ \mathbf{d}^\top \mathbf{y} = 1}} 2\mathbf{d}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{R}^{-1} \mathbf{y}.$$

The reason is that for any feasible \mathbf{y} of (33), the vector $\mathbf{y}' := \mathbf{y}/\mathbf{d}^\top \mathbf{y}$ is a feasible solution of this new program, and in particular the optimal solution \mathbf{y}^* of (33) induces a solution $\mathbf{y}^*/\mathbf{d}^\top \mathbf{y}^* = \mathbf{y}^*/\Psi(\mathbf{r})$ for this new program, and it gives an objective value of $2 - \frac{1}{\Psi(\mathbf{r})}$. Now it suffices to prove that $\mathbf{y}^*/\mathbf{d}^\top \mathbf{y}^*$ is the optimal solution of this new program. Suppose by contrary that there is another vector \mathbf{y}' that is the optimal solution of this new program, then we have $\mathbf{d}^\top \mathbf{y}' = 1$ and $\mathbf{y}'^\top \mathbf{R}^{-1} \mathbf{y}' \leq 1/\Psi(\mathbf{r})$. But then the vector $\mathbf{y}' \cdot \Psi(\mathbf{r})$ would be a solution for (33) which gives a larger objective value than $\Psi(\mathbf{r})$, and this contradicts with the definition that $\Psi(\mathbf{r})$ is the optimal objective value of (33).

We can now write the following problem,

$$\frac{1}{\Psi(\mathbf{r})} = \min_{\substack{\mathbf{C}^\top \mathbf{y}=0 \\ \mathbf{d}^\top \mathbf{y}=1}} \mathbf{y}^\top \mathbf{R}^{-1} \mathbf{y}, \quad (35)$$

with optimum value at $\tilde{\mathbf{y}} = \frac{\mathbf{y}^*}{\Psi(\mathbf{r})}$. Let us now consider Program 35 at \mathbf{r}' , i.e., $\Psi(\mathbf{r}')$. We note that $\tilde{\mathbf{y}} = \frac{\mathbf{y}^*}{\Psi(\mathbf{r})}$ is a feasible solution for the new program as well. Therefore,

$$\frac{1}{\Psi(\mathbf{r}')} = \min_{\substack{\mathbf{C}^\top \mathbf{y}=0 \\ \mathbf{d}^\top \mathbf{y}=1}} \mathbf{y}^\top \mathbf{R}'^{-1} \mathbf{y} \leq \frac{1}{\Psi(\mathbf{r})^2} \mathbf{y}^{*\top} \mathbf{R}'^{-1} \mathbf{y}^* = \frac{1}{\Psi(\mathbf{r})^2} \sum_e \frac{\mathbf{r}_e}{\mathbf{r}'_e} \mathbf{r}_e (\mathbf{C} \Delta^* - \mathbf{d})_e^2.$$

Rearranging the above,

$$\frac{1}{\Psi(\mathbf{r}')} \leq \frac{1}{\Psi(\mathbf{r})} \left(1 - \frac{\sum_e \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}'_e} \right) \mathbf{r}_e (\mathbf{C} \Delta^* - \mathbf{d})_e^2}{\Psi(\mathbf{r})} \right).$$

And since we assumed that $|\mathbf{r}'_e - \mathbf{r}_e| \leq \mathbf{r}'_e$, we have that $\sum_e \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}'_e} \right) \mathbf{r}_e (\mathbf{C} \Delta^* - \mathbf{d})_e^2 \leq \Psi(\mathbf{r})$, and

hence $\left(1 - \frac{\sum_e \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}'_e} \right) \mathbf{r}_e (\mathbf{C} \Delta^* - \mathbf{d})_e^2}{\Psi(\mathbf{r})} \right)^{-1} \geq \left(1 + \frac{\sum_e \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}'_e} \right) \mathbf{r}_e (\mathbf{C} \Delta^* - \mathbf{d})_e^2}{\Psi(\mathbf{r})} \right)$, and so we have

$$\Psi(\mathbf{r}') \geq \Psi(\mathbf{r}) + \sum_e \left(\frac{\mathbf{r}'_e - \mathbf{r}_e}{\mathbf{r}'_e} \right) \mathbf{r}_e (\mathbf{C} \Delta^* - \mathbf{d})_e^2.$$

□

Proof of Fact A.6

Proof. Let $x = \alpha \cdot p + (1 - \alpha) \cdot q$ for $\alpha \in (0, 1)$. For notational simplicity, we assume that n^β , n^p , n^q , and n^α are all integers. Consider two rectangular matrices of dimensions $n \times n^x$ and $n^x \times n^\beta$. Since $\alpha p \leq x$, we can tile the $n \times n^x$ rectangular matrix with matrices of dimensions $n^\alpha \times n^{\alpha p}$, and tile the $n^x \times n^\beta$ rectangular matrix with matrices of dimensions $n^{\alpha p} \times n^{\alpha \beta}$. Then, the product of the two tiled matrices can be obtained by viewing it as a multiplication of a matrix of dimensions $n/n^\alpha \times n^x/n^{\alpha p}$ with a matrix of dimensions $n^x/n^{\alpha p} \times n^\beta/n^{\alpha \beta}$, where each “element” of the first matrix is itself a matrix of dimensions $n^\alpha \times n^{\alpha p}$, and each “element” of the second matrix is itself a matrix of dimensions $n^{\alpha p} \times n^{\alpha \beta}$. With this recursion in tow, we obtain the following upper bound.

$$\begin{aligned} \mathcal{T}_{\text{mat}}(n, n^x, n^\beta) &\leq \mathcal{T}_{\text{mat}}(n^\alpha, n^{\alpha p}, n^{\alpha \beta}) \cdot \mathcal{T}_{\text{mat}}(n/n^\alpha, n^x/n^{\alpha p}, n^\beta/n^{\alpha \beta}) \\ &= \mathcal{T}_{\text{mat}}(n^\alpha, n^{\alpha p}, n^{\alpha \beta}) \cdot \mathcal{T}_{\text{mat}}(n^{(1-\alpha)}, n^{(1-\alpha)q}, n^{(1-\alpha)\beta}) \\ &\leq n^{\alpha \cdot \omega_\beta(p) + o(1)} \cdot n^{(1-\alpha) \cdot \omega_\beta(q) + o(1)}, \end{aligned}$$

where the last step follows from denoting $m_1 = n^\alpha$ and observing that by the definition of $\omega_\beta(x)$, multiplying matrices of dimensions $n^\alpha \times n^{\alpha p}$ and $n^{\alpha p} \times n^{\alpha \beta}$ costs $m_1^{\omega_\beta(p) + o(1)}$, which is exactly $n^{\alpha(\omega_\beta(p) + o(1))}$, and similarly denoting $m_2 = n^{1-\alpha}$, we have that multiplying matrices of dimensions $n^{1-\alpha} \times n^{(1-\alpha)q}$ and $n^{(1-\alpha)q} \times n^{(1-\alpha)\beta}$ costs $m_2^{\omega_\beta(q) + o(1)}$, which is exactly $n^{(1-\alpha)(\omega_\beta(q) + o(1))}$. Comparing exponents, this implies that

$$\omega_\beta(x) \leq \alpha \cdot \omega_\beta(p) + (1 - \alpha) \cdot \omega_\beta(q),$$

which proves the convexity of the function $\omega_\beta(x)$. □

Missing Proofs of inverse maintenance data structures

Proof of Lemma C.1

Proof. The initialization and update operations are straightforward. Next we prove the time complexity of the reset and query operations.

Consider the t -th iteration. We define $\Delta := \Delta^{(t_0+1)} + \dots + \Delta^{(t)}$, and note that $k = \text{nnz}(\Delta^{(t_0+1)}) + \dots + \text{nnz}(\Delta^{(t)}) \geq \text{nnz}(\Delta)$. W.l.o.g. we assume $k = \text{nnz}(\Delta)$. We write the decomposition $\Delta = \mathbf{U} \mathbf{C} \mathbf{V}^\top$, where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{n \times k}$ each consists of k columns of the identity matrix, and $\mathbf{C} \in \mathbb{R}^{k \times k}$ consists of the non-zero entries of Δ .

Reset. Using Woodbury identity, we have

$$\begin{aligned} (\mathbf{M}^{(t)})^{-1} &= (\mathbf{M}^{(t_0)} + \Delta)^{-1} \\ &= (\mathbf{M}^{(t_0)} + \mathbf{U} \mathbf{C} \mathbf{V}^\top)^{-1} \\ &= (\mathbf{M}^{(t_0)})^{-1} - (\mathbf{M}^{(t_0)})^{-1} \mathbf{U} (\mathbf{C}^{-1} + \mathbf{V}^\top (\mathbf{M}^{(t_0)})^{-1} \mathbf{U})^{-1} \mathbf{V}^\top (\mathbf{M}^{(t_0)})^{-1} \\ &= \mathbf{N} - \mathbf{N} \mathbf{U} (\mathbf{C}^{-1} + \mathbf{V}^\top \mathbf{N} \mathbf{U})^{-1} \mathbf{V}^\top \mathbf{N}. \end{aligned}$$

The matrices $\mathbf{N} \mathbf{U}$ and $\mathbf{V}^\top \mathbf{N}$ can be directly read off from the maintained inverse \mathbf{N} . The dominating term to compute this inverse is to multiply the $n \times k$ matrix $\mathbf{N} \mathbf{U}$ with the $k \times n$ matrix $(\mathbf{C}^{-1} + \mathbf{V}^\top \mathbf{N} \mathbf{U})^{-1} \mathbf{V}^\top \mathbf{N}$, and it takes $\mathcal{T}_{\text{mat}}(n, n, k)$ time.

Query. Again using Woodbury identity, we have

$$((\mathbf{M}^{(t)})^{-1})_{J_r, J_c} = \mathbf{N}_{J_r, J_c} - \mathbf{N}_{J_r, :} \mathbf{U} (\mathbf{C}^{-1} + \mathbf{V}^\top \mathbf{N} \mathbf{U})^{-1} \mathbf{V}^\top \mathbf{N}_{:, J_c}.$$

W.l.o.g. we assume $\ell_c \leq \ell_r$. The running time has the following parts:

- Computing the inverse $(\mathbf{C}^{-1} + \mathbf{V}^\top \mathbf{N} \mathbf{U})^{-1}$ takes k^ω time.
- Computing $(\mathbf{C}^{-1} + \mathbf{V}^\top \mathbf{N} \mathbf{U})^{-1} \cdot (\mathbf{V}^\top \mathbf{N}_{:, J_c})$ takes $\mathcal{T}_{\text{mat}}(k, k, \ell_c)$ time.
- Computing $(\mathbf{N}_{J_r, :} \mathbf{U}) \cdot (\mathbf{C}^{-1} + \mathbf{V}^\top \mathbf{N} \mathbf{U})^{-1} \mathbf{V}^\top \mathbf{N}_{:, J_c}$ takes $\mathcal{T}_{\text{mat}}(\ell_r, k, \ell_c)$ time.

Since if $\ell_c \leq k$, then $\mathcal{T}_{\text{mat}}(k, k, \ell_c) \leq k^\omega$, and otherwise if $\ell_c > k$, then $\mathcal{T}_{\text{mat}}(k, k, \ell_c) \leq \mathcal{T}_{\text{mat}}(\ell_r, k, \ell_c)$, so the total time is

$$O(k^\omega + \mathcal{T}_{\text{mat}}(\ell_r, k, \ell_c)).$$

□

Proof of Lemma C.2

Proof. The initialization and update operations are straightforward. Next we prove the time complexity of the reset, partial reset, and query operations.

Consider the t -th iteration. We define $\Delta_0 := \Delta^{(t_0+1)} + \dots + \Delta^{(t)}$, and note that $k_0 = \text{nnz}(\Delta^{(t_0+1)}) + \dots + \text{nnz}(\Delta^{(t)}) \geq \text{nnz}(\Delta_0)$. Similarly define $\Delta_1 := \Delta^{(t_1+1)} + \dots + \Delta^{(t)}$ and note that $k_1 \geq \text{nnz}(\Delta_1)$.

Reset. We compute $(\mathbf{M}^{(t)})^{-1} = (\mathbf{M}^{(t_0)} + \Delta_0)^{-1}$ using Woodbury identity. Similar to the reset operation of Lemma C.1, it takes $O(\mathcal{T}_{\text{mat}}(n, n, k_0))$ time.

Partial reset. Let $J^{\text{new}} \subseteq [n]$ denote the indexes of the non-zero columns of $\Delta^{(t_0+1)} + \dots + \Delta^{(t)}$ and let $\mathbf{B}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1}$ and $\mathbf{E}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :}$; denote the matrices that we want to obtain. Next we show how to compute these two matrices efficiently.

First note that we have

$$\begin{aligned}
\mathbf{T}^{(t_0,t)} &= \mathbf{I} + (\mathbf{M}^{(t_0)})^{-1} \cdot (\mathbf{M}^{(t)} - \mathbf{M}^{(t_0)}) \\
&= \mathbf{I} + (\mathbf{M}^{(t_0)})^{-1} \cdot (\mathbf{M}^{(t_1)} - \mathbf{M}^{(t_0)}) + (\mathbf{M}^{(t_0)})^{-1} \cdot (\mathbf{M}^{(t)} - \mathbf{M}^{(t_1)}) \\
&= \mathbf{T}^{(t_0,t_1)} + \mathbf{N} \cdot \Delta_1.
\end{aligned}$$

Note that $\mathbf{T}^{(t_0,t_1)}$ is identity matrix plus some non-zero entries on the columns in J , so we have

$$\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0,t)} = \begin{bmatrix} \mathbf{T}_{J,J}^{(t_0,t_1)} & 0 \\ 0 & \mathbf{I} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \mathbf{T}_{J^{\text{new}} \setminus J, J}^{(t_0,t_1)} & 0 \end{bmatrix} + (\mathbf{N} \cdot \Delta_1)_{J^{\text{new}}, J^{\text{new}}}$$

Since $\text{nnz}(\mathbf{M}^{(t_1)} - \mathbf{M}^{(t_0)}) \leq k_0$ and $|J^{\text{new}} \setminus J| \leq k_1$, we can compute $\mathbf{T}_{J^{\text{new}} \setminus J, J}^{(t_0,t_1)} = (\mathbf{N} \cdot (\mathbf{M}^{(t_1)} - \mathbf{M}^{(t_0)}))_{J^{\text{new}} \setminus J, J}$ in $O(k_0 k_1)$ time. Since $\text{nnz}(\Delta_1) \leq k_1$, let $J' \subseteq$ denote the row indexes of the non-zero entries of Δ_1 , we can write $(\mathbf{N} \cdot \Delta_1)_{J^{\text{new}}, J^{\text{new}}} = (\mathbf{N}_{J^{\text{new}}, J'}) \cdot (\Delta_1)_{J', J^{\text{new}}}$ in $O(k_0 k_1)$ time. So in conclusion, we have

$$\begin{aligned}
\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0,t)} &= \begin{bmatrix} \mathbf{T}_{J,J}^{(t_0,t_1)} & 0 \\ 0 & \mathbf{I} \end{bmatrix} + (\mathbf{I}_{J^{\text{new}}, J^{\text{new}} \setminus J}) \cdot (\mathbf{T}_{J^{\text{new}} \setminus J, J^{\text{new}}}^{(t_0,t_1)}) + (\mathbf{N}_{J^{\text{new}}, J'}) \cdot (\Delta_1)_{J', J^{\text{new}}} \\
&= \begin{bmatrix} \mathbf{T}_{J,J}^{(t_0,t_1)} & 0 \\ 0 & \mathbf{I} \end{bmatrix} + \mathbf{U} \mathbf{V}^\top,
\end{aligned}$$

where both \mathbf{U} and \mathbf{V} have size at most $k_0 \times O(k_1)$, and we can compute \mathbf{U} and \mathbf{V} in $O(k_0 k_1)$ time. Furthermore, because $\mathbf{U} = [(\mathbf{I}_{J^{\text{new}}, J^{\text{new}} \setminus J}), (\mathbf{N}_{J^{\text{new}}, J'})]$ and since we already maintain $\mathbf{E} = \mathbf{B} \cdot \mathbf{N}_{J,:}$, we directly read off entries from the matrix $\begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \cdot \mathbf{U}$.

Then using Woodbury identity and since we already have $\mathbf{B} = (\mathbf{T}_{J,J}^{(t_0,t_1)})^{-1}$, we can compute the updated $\mathbf{B}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0,t)})^{-1}$ in $O(\mathcal{T}_{\text{mat}}(k_0, k_0, k_1))$ time.

Next we compute the updated \mathbf{E}^{new} using the maintained $\mathbf{E} = (\mathbf{T}_{J,J}^{(t_0,t_1)})^{-1} \cdot \mathbf{N}_{J,:}$. Using Woodbury identity, we have

$$\begin{aligned}
\mathbf{E}^{\text{new}} &= \left(\begin{bmatrix} \mathbf{T}_{J,J}^{(t_0,t_1)} & 0 \\ 0 & \mathbf{I} \end{bmatrix} + \mathbf{U} \mathbf{V}^\top \right)^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :} \\
&= \left(\begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} - \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U} (\mathbf{I} + \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U})^{-1} \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \right) \cdot \mathbf{N}_{J^{\text{new}}, :} \\
&= \begin{bmatrix} \mathbf{E} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \end{bmatrix} - \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U} (\mathbf{I} + \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U})^{-1} \mathbf{V}^\top \begin{bmatrix} \mathbf{E} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \end{bmatrix}. \tag{36}
\end{aligned}$$

The dominating terms are to compute $\mathbf{V}^\top \cdot \begin{bmatrix} \mathbf{E} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \end{bmatrix}$ and to multiply $\begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U}$ with the matrix $(\mathbf{I} + \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U})^{-1} \mathbf{V}^\top \begin{bmatrix} \mathbf{E} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \end{bmatrix}$, and both of these two steps take $O(\mathcal{T}_{\text{mat}}(n, k_0, k_1))$ time.

In summary, the total time of the partial reset operation is $O(\mathcal{T}_{\text{mat}}(n, k_0, k_1))$.

Query. By the definition of the transformation matrix, we have $\mathbf{M}^{(t)} = \mathbf{M}^{(t_0)} \cdot \mathbf{T}^{(t_0, t)}$. Again let $J^{\text{new}} \subseteq [n]$ denote the indexes of the non-zero columns of $\Delta^{(t_0+1)}, \dots, \Delta^{(t)}$. And recall that $J \subseteq [n]$ denotes the indexes of the non-zero columns of $\Delta^{(t_0+1)}, \dots, \Delta^{(t_1)}$.

Using a similar computation as the partial update operation, we can write $\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)} = \begin{bmatrix} \mathbf{T}_{J, J}^{(t_0, t_1)} & 0 \\ 0 & \mathbf{I} \end{bmatrix} + \mathbf{U} \mathbf{V}^\top$, where both \mathbf{U} and \mathbf{V} have size $k_0 \times O(k_1)$, and we can compute them in $O(k_0 k_1)$ time. Again denote $\mathbf{E}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :}$, and it can be written as Eq. (36). In the query operation we don't compute the matrix \mathbf{E}^{new} explicitly, instead, we note that by using Eq. (36) and since the matrix $\begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \cdot \mathbf{U}$ is already maintained, we can compute any $\ell_r \times \ell_c$ submatrix of \mathbf{E}^{new} in $O(\mathcal{T}_{\text{mat}}(k_0, k_1, k_1) + \mathcal{T}_{\text{mat}}(k_0, k_1, \ell_c) + \mathcal{T}_{\text{mat}}(\ell_r, k_1, \ell_c))$ time.

Next note that since $\mathbf{M}^{(t)}$ only differs from $\mathbf{M}^{(t_0)}$ on columns in set J^{new} , denoting $\overline{J^{\text{new}}} = [n] \setminus J^{\text{new}}$, we can write $\mathbf{T}^{(t_0, t)}$ as

$$\mathbf{T}^{(t_0, t)} = \begin{bmatrix} \mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)} & 0 \\ \mathbf{T}_{\overline{J^{\text{new}}}, J^{\text{new}}}^{(t_0, t)} & \mathbf{I} \end{bmatrix}.$$

So its inverse is $\begin{bmatrix} (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} & 0 \\ -\mathbf{T}_{\overline{J^{\text{new}}}, J^{\text{new}}}^{(t_0, t)} \cdot (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} & \mathbf{I} \end{bmatrix}$, and we have

$$\begin{aligned} (\mathbf{M}^{(t)})^{-1} &= (\mathbf{T}^{(t_0, t)})^{-1} \cdot (\mathbf{M}^{(t_0)})^{-1} \\ &= \begin{bmatrix} (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} & 0 \\ -\mathbf{T}_{\overline{J^{\text{new}}}, J^{\text{new}}}^{(t_0, t)} \cdot (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} & \mathbf{I} \end{bmatrix} \cdot \mathbf{N} \\ &= \begin{bmatrix} (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :} \\ -(\mathbf{I} + \mathbf{N} \cdot \Delta_0)_{\overline{J^{\text{new}}}, J^{\text{new}}} \cdot (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :} + \mathbf{N}_{\overline{J^{\text{new}}}, :} \end{bmatrix}. \end{aligned} \quad (37)$$

To compute a $\ell_r \times \ell_c$ submatrix of $(\mathbf{M}^{(t)})^{-1}$, we first compute ℓ_c columns of $(\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :}$, and as we just proved, this takes $O(\mathcal{T}_{\text{mat}}(k_0, k_1, k_1) + \mathcal{T}_{\text{mat}}(k_0, k_1, \ell_c))$ time. Next we compute the multiplication of ℓ_r rows of $(\mathbf{N} \cdot \Delta_0)_{\overline{J^{\text{new}}}, J^{\text{new}}}$ with ℓ_c columns of $(\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :}$, and this takes $O(\ell_r, k_0, \ell_c)$.

In summary, the total time of the query operation is $O(\mathcal{T}_{\text{mat}}(k_0, k_1, k_1) + \mathcal{T}_{\text{mat}}(k_0, k_1, \ell_c) + \mathcal{T}_{\text{mat}}(k_0, \ell_r, \ell_c))$. \square

Proof of Lemma C.4

Proof. In the algorithm we always maintain the following invariants:

$$\begin{aligned} J &= \text{set of indexes of the non-zero columns of } \Delta^{(t_0+1)} + \dots + \Delta^{(t_1)} \\ \mathbf{N} &= (\mathbf{M}^{(t_0)})^{-1} \\ \mathbf{B} &= (\mathbf{T}_{J, J}^{(t_0, t_1)})^{-1} \\ \mathbf{E} &= (\mathbf{T}_{J, J}^{(t_0, t_1)})^{-1} \cdot \mathbf{N}_{J, :} \\ \mathbf{u}_0 + \mathbf{N} \mathbf{u}_1 + \begin{bmatrix} \mathbf{N}_{J, :} \cdot \mathbf{u}_2 \\ 0 \end{bmatrix} &= \sum_{i=0}^t (\mathbf{M}^{(i)})^{-1} \mathbf{v}. \end{aligned}$$

Apart from these invariants, the data structure also maintains the vectors $\mathbf{N} \cdot \mathbf{v}$, $\mathbf{B} \cdot \mathbf{v}_J$, and $\mathbf{E} \cdot \mathbf{v}$.

Next we describe each operation and bound its time complexity.

Initialize. Initially we let $t_0 = t_1 = 0$. Let $\mathbf{N} = (\mathbf{M}^{(0)})^{-1} \in \mathbb{R}^{n \times n}$, and let \mathbf{B} and \mathbf{E} be empty matrices. We also pre-compute and maintain $\mathbf{N} \cdot \mathbf{v}$. We let $\mathbf{u}_0 = (\mathbf{M}^{(0)})^{-1} \cdot \mathbf{v}$ and $\mathbf{u}_1 = \mathbf{u}_2 = 0$. Initialization takes $O(n^\omega)$ time.

Query sum. Since we maintain the invariants, we simply output $\mathbf{u}_0 + \mathbf{N}\mathbf{u}_1 + \begin{bmatrix} \mathbf{N}_{J,:} \cdot \mathbf{u}_2 \\ 0 \end{bmatrix} = \sum_{i=0}^t (\mathbf{M}^{(i)})^{-1} \mathbf{v}$, and this takes $O(n^2)$ time.

For the reset, partial reset, and update operations, we define the following notations. Consider the t -th iteration. We define $\Delta_0 := \Delta^{(t_0+1)} + \dots + \Delta^{(t)}$, and note that $k_0 = \text{nnz}(\Delta^{(t_0+1)}) + \dots + \text{nnz}(\Delta^{(t)}) \geq \text{nnz}(\Delta_0)$. Similarly define $\Delta_1 := \Delta^{(t_1+1)} + \dots + \Delta^{(t)}$ and note that $k_1 \geq \text{nnz}(\Delta_1)$.

Reset. We first update $\mathbf{u}_0 \leftarrow \mathbf{u}_0 + \mathbf{N}\mathbf{u}_1 + \begin{bmatrix} \mathbf{N}_{J,:} \cdot \mathbf{u}_2 \\ 0 \end{bmatrix}$ and $\mathbf{u}_1, \mathbf{u}_2 \leftarrow 0$.

Next, we update $\mathbf{N} = (\mathbf{M}^{(t)})^{-1} = (\mathbf{M}^{(t_0)} + \Delta_0)^{-1}$ using Woodbury identity in the exact same way as the reset operation of Lemma C.1 and C.2, and this operation takes $O(\mathcal{T}_{\text{mat}}(n, n, k_0))$ time.

Finally we also recompute the vector $\mathbf{N} \cdot \mathbf{v}$ in $O(n^2)$ time.

Partial reset. We first update $\mathbf{u}_0 \leftarrow \mathbf{u}_0 + \begin{bmatrix} \mathbf{N}_{J,:} \cdot \mathbf{u}_2 \\ 0 \end{bmatrix}$ and $\mathbf{u}_2 \leftarrow 0$. This takes $O(n \cdot k_0)$ time.

Next let $J^{\text{new}} \subseteq [n]$ denote the indexes of the non-zero columns of $\Delta^{(t_0+1)} + \dots + \Delta^{(t)}$ and let $\mathbf{B}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1}$ and $\mathbf{E}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :}$ denote the matrices that we want to obtain. We compute \mathbf{B}^{new} and \mathbf{E}^{new} in the exact same way as Lemma C.2 in $O(\mathcal{T}_{\text{mat}}(n, k_0, k_1))$ time.

Finally we also recompute the vectors $\mathbf{B}^{\text{new}} \cdot \mathbf{v}_{J^{\text{new}}}$ and $\mathbf{E}^{\text{new}} \cdot \mathbf{v}$ in $O(n \cdot k_0)$ time.

Update. Again let $J^{\text{new}} \subseteq [n]$ denote the indexes of the non-zero columns of $\Delta^{(t_0+1)} + \dots + \Delta^{(t)}$ and let $\mathbf{B}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1}$ and $\mathbf{E}^{\text{new}} = (\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)})^{-1} \cdot \mathbf{N}_{J^{\text{new}}, :}$. Similar as the proof of Lemma C.2, we can compute the decomposition $\mathbf{T}_{J^{\text{new}}, J^{\text{new}}}^{(t_0, t)} = \begin{bmatrix} \mathbf{T}_{J, J}^{(t_0, t_1)} & 0 \\ 0 & \mathbf{I} \end{bmatrix} + \mathbf{U} \mathbf{V}^\top$ and $\begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \cdot \mathbf{U}$ in $O(k_0 k_1)$ time, where both \mathbf{U} and \mathbf{V} have size $k_0 \times O(k_1)$. And we still have Eq. (37) and (36) that

$$(\mathbf{M}^{(t)})^{-1} = \begin{bmatrix} \mathbf{E}^{\text{new}} \\ -(\mathbf{I} + \mathbf{N} \cdot \Delta_0)_{J^{\text{new}}, J^{\text{new}}} \cdot \mathbf{E}^{\text{new}} + \mathbf{N}_{J^{\text{new}}, :} \end{bmatrix},$$

$$\mathbf{E}^{\text{new}} = \begin{bmatrix} \mathbf{E} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \end{bmatrix} - \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U} (\mathbf{I} + \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U})^{-1} \mathbf{V}^\top \begin{bmatrix} \mathbf{E} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \end{bmatrix}.$$

So we have

$$\mathbf{E}^{\text{new}} \mathbf{v} = \begin{bmatrix} \mathbf{E} \mathbf{v} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \mathbf{v} \end{bmatrix} - \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U} (\mathbf{I} + \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U})^{-1} \mathbf{V}^\top \begin{bmatrix} \mathbf{E} \mathbf{v} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \mathbf{v} \end{bmatrix}$$

We can compute this vector in the following steps:

- Since the data structure maintains $\mathbf{E}\mathbf{v}$ and $\mathbf{N}\mathbf{v}$, we first directly read off the vector $\begin{bmatrix} \mathbf{E}\mathbf{v} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \mathbf{v} \end{bmatrix}$ from the maintained vectors.

- Compute $(\mathbf{I} + \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U})^{-1}$ in $O(\mathcal{T}_{\text{mat}}(k_1, k_0, k_1))$ time.

- Compute the matrix vector products from right to left as

$$\left(\begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U} \right) \cdot (\mathbf{I} + \mathbf{V}^\top \begin{bmatrix} \mathbf{B} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \mathbf{U})^{-1} \cdot \mathbf{V}^\top \cdot \begin{bmatrix} \mathbf{E}\mathbf{v} \\ \mathbf{N}_{J^{\text{new}} \setminus J, :} \mathbf{v} \end{bmatrix},$$

and this takes $O(k_0 k_1)$ time.

In summary we can compute $\mathbf{E}^{\text{new}} \mathbf{v}$ in $O(\mathcal{T}_{\text{mat}}(k_0, k_1, k_1))$ time. Next we compute $(\mathbf{M}^{(t)})^{-1} \cdot \mathbf{v}$:

$$\begin{aligned} (\mathbf{M}^{(t)})^{-1} \cdot \mathbf{v} &= \begin{bmatrix} \mathbf{E}^{\text{new}} \mathbf{v} \\ -(\mathbf{I} + \mathbf{N} \cdot \Delta_0)_{\overline{J^{\text{new}}}, J^{\text{new}}} \cdot \mathbf{E}^{\text{new}} \mathbf{v} + \mathbf{N}_{\overline{J^{\text{new}}}, :} \mathbf{v} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}^{\text{new}} \mathbf{v} \\ -\mathbf{I}_{\overline{J^{\text{new}}}, J^{\text{new}}} \cdot \mathbf{E}^{\text{new}} \mathbf{v} + \mathbf{N}_{\overline{J^{\text{new}}}, :} \cdot (\mathbf{v} - \mathbf{w}) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{E}^{\text{new}} \mathbf{v} - \mathbf{N}_{J^{\text{new}}, :} \cdot (\mathbf{v} - \mathbf{w}) \\ -\mathbf{I}_{\overline{J^{\text{new}}}, J^{\text{new}}} \cdot \mathbf{E}^{\text{new}} \mathbf{v} \end{bmatrix} + \mathbf{N} \cdot (\mathbf{v} - \mathbf{w}) \\ &= \begin{bmatrix} \mathbf{E}^{\text{new}} \mathbf{v} - \mathbf{w}' \\ -\mathbf{I}_{\overline{J^{\text{new}}}, J^{\text{new}}} \cdot \mathbf{E}^{\text{new}} \mathbf{v} \end{bmatrix} - \begin{bmatrix} \mathbf{N}_{J, :} \cdot (\mathbf{v} - \mathbf{w}) \\ 0 \end{bmatrix} + \mathbf{N} \cdot (\mathbf{v} - \mathbf{w}), \end{aligned}$$

where in the second step we define a vector $\mathbf{w} \in \mathbb{R}^n$ such that its entries in J^{new} are $(\Delta_0)_{J^{\text{new}}, J^{\text{new}}} \cdot \mathbf{E}^{\text{new}} \mathbf{v}$ and its rest entries are all zero, and we can compute it in $O(k_0)$ time since $\text{nnz}(\Delta_0) \leq k_0$, in the fourth step we define a vector $\mathbf{w}' \in \mathbb{R}^{|J^{\text{new}}|}$ such that its entries in $J^{\text{new}} \setminus J$ are $\mathbf{N}_{J^{\text{new}} \setminus J, :} \cdot (\mathbf{v} - \mathbf{w})$ and its rest entries are all zero, and we can compute it in $O(k_0 k_1)$ time since $|J^{\text{new}} \setminus J| \leq k_1$.

So we update the vectors as

$$\mathbf{u}_0 \leftarrow \mathbf{u}_0 + \begin{bmatrix} \mathbf{E}^{\text{new}} \mathbf{v} - \mathbf{w}' \\ -\mathbf{I}_{\overline{J^{\text{new}}}, J^{\text{new}}} \cdot \mathbf{E}^{\text{new}} \mathbf{v} \end{bmatrix}, \quad \mathbf{u}_1 \leftarrow \mathbf{u}_1 + \mathbf{v} - \mathbf{w}, \quad \mathbf{u}_w \leftarrow \mathbf{u}_1 - \mathbf{v} + \mathbf{w}.$$

In this way we still maintain the invariant of the three vectors. \square

J MWU with Non-Monotone Weights and $n^{1/3}$ Iterations (Optional)

In this section, we will prove Theorem G.1. Our analysis follows a similar structure to that of Algorithm 1. We would track the same potentials as defined in Equations (3) and (4).

In this section, we again use the notation k_i to denote the total number of width reduction steps taken before the i^{th} primal step is being executed, and we use the notation i_k to denote the number of primal steps taken by the algorithm when the k^{th} width step is being executed. We show that for input $\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} \\ -\mathbf{C} \end{bmatrix}$ and $\tilde{\mathbf{d}} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix}$, the algorithm returns $\tilde{\mathbf{x}}$ such that $\|\mathbf{C}\tilde{\mathbf{x}} - \mathbf{d}\|_\infty \leq 1 + O(\epsilon)$.

Convergence Analysis

We begin by showing that $\mathbf{w}^{(i,k)} > 0$ for all i and k . Observe that this was always true for Algorithm 1.

Lemma J.1. *If $\alpha\rho^{1/3} \leq \frac{\epsilon^{1/3}}{10n^{1/3}}$ and $\delta \leq 1/2$, then for every iteration i and k and every coordinate e , $\mathbf{w}_e^{(i,k)} > 0$. Furthermore, every primal update satisfies, $|\vec{\alpha}^{(i,k)}(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})| \leq 1/10$.*

Proof. The weights will always increase during a width reduction step and can only decrease during a primal step. Therefore, we prove that after a primal step the weights can never be negative, i.e., for every i, k , $1 + \epsilon\vec{\alpha}(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}) \geq \frac{1}{2}$.

Observe that, when we do a primal step, i.e., the condition on Line 11 of Algorithm 10 is true, since for all e , $\mathbf{r}_e^{(i,k)} \geq \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k)}) \geq \frac{\epsilon}{2n}\Psi(\mathbf{r}^{(i,k)})$,

$$\frac{\epsilon}{2n}\Psi(\mathbf{r}^{(i,k)})\|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}\|_3^3 \leq 2\rho\Psi(\mathbf{r}^{(i,k)}).$$

This implies that,

$$\|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}\|_\infty \leq \|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}\|_3 \leq \frac{4^{1/3}n^{1/3}\rho^{1/3}}{\epsilon^{1/3}} \leq \frac{2n^{1/3}\rho^{1/3}}{\epsilon^{1/3}}.$$

When $(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e \geq 0$, our weights can only increase, therefore we consider the case when it is negative. The multiplicative change to the weights now becomes,

$$1 + \epsilon\vec{\alpha}(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}) \geq 1 - \epsilon\alpha_- \|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}\|_\infty \geq 1 - \frac{\epsilon}{1+2\epsilon}\alpha \cdot \frac{4n^{1/3}\rho^{1/3}}{\epsilon^{1/3}} \geq \frac{1}{2}.$$

The second part just follows by using the values of $\vec{\alpha}^{(i,k)}$ and the bound on $\|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}\|_\infty$. **We would like to remark that this proof also works for the further robust algorithm described in Section G.2** \square

We now begin our analysis. The next two lemmas show how our potentials change with every iteration of the algorithm.

Change in Φ

Lemma J.2. *After i primal steps, and k width-reduction steps, the potential Φ is bounded as follows:*

$$\Phi(\mathbf{w}^{(i,k)}) \leq \left(\Phi(\mathbf{w}^{(0,0)})\right)\left(1 + \epsilon\alpha_+e^{\epsilon/2}\right)^i\left(1 + \frac{2\epsilon e^\epsilon}{\tau}\right)^k.$$

Furthermore, after every primal step, the potential can decrease by at most,

$$\Phi(\mathbf{w}^{(i+1,k)}) \geq \Phi(\mathbf{w}^{(i,k)})\left(1 - \epsilon\alpha_+e^{\epsilon/2}\right)^i.$$

Proof. We prove this claim by induction. Initially, $i = k = 0$, and the claim holds trivially. Assume that the claim holds for some $i, k \geq 0$. We will use Φ as an abbreviated notation for $\Phi(\mathbf{w}^{(i,k)})$, $\vec{\alpha}$ for $\vec{\alpha}^{(i,k)}$, and \mathbf{w} to denote $\mathbf{w}^{(i,k)}$.

Primal Step. If the next step is a *primal* step,

$$\Phi(\mathbf{w}^{(i+1,k)}) = \left\| \mathbf{w} + \epsilon \vec{\alpha} (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}}) \mathbf{w} \right\|_1 = \|\mathbf{w}\|_1 + \epsilon \sum_e \mathbf{w}_e \vec{\alpha}_e (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e \quad (38)$$

We first bound $\sum_e \mathbf{w}_e \cdot \vec{\alpha}_e \cdot (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e$. Using Cauchy-Schwarz inequality, we have

$$\begin{aligned} \sum_e \mathbf{w}_e \cdot \vec{\alpha}_e \cdot |\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}}|_e &\leq \sqrt{\left(\sum_e \mathbf{w}_e (\vec{\alpha}_e)^2 \right) \cdot \left(\sum_e \mathbf{w}_e \cdot (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e^2 \right)} \\ &\leq \alpha_+ \cdot \sqrt{\Phi(\mathbf{w}) \cdot \Psi(\mathbf{r})} \\ &\leq \alpha_+ \cdot e^{\epsilon/2} \cdot \Phi(\mathbf{w}), \end{aligned}$$

where the second step follows from $\alpha_- < \alpha_+$, the third step follows from Lemma A.1 that $\Psi(\mathbf{r}) \leq e^\epsilon \cdot \Phi(\mathbf{w})$.

Now, from Equation (38), and the fact that $\mathbf{w}_e > 0$ from Lemma J.1,

$$\Phi(\mathbf{w}) - \epsilon \sum_e \mathbf{w}_e \vec{\alpha}_e |\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}}|_e \leq \Phi(\mathbf{w}^{(i+1,k)}) \leq \Phi(\mathbf{w}) + \epsilon \sum_e \mathbf{w}_e \vec{\alpha}_e |\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}}|_e.$$

Therefore, we get our bounds,

$$\Phi(\mathbf{w}^{(i,k)}) (1 - \epsilon \alpha_+ e^{\epsilon/2}) \leq \Phi(\mathbf{w}^{(i+1,k)}) \leq \Phi(\mathbf{w}^{(i,k)}) (1 + \epsilon \alpha_+ e^{\epsilon/2})$$

Width Reduction Step. Let Δ be the solution returned in Line 9 of Algorithm 10. We have the following:

$$\begin{aligned} \Phi(\mathbf{w}^{(i,k+1)}) &= \sum_{e \notin H \cup \{\bar{e}\}} \mathbf{w}_e^{(i,k)} + \sum_{e \in H} \left((1 + \epsilon) \mathbf{w}_e^{(i,k)} + \frac{\epsilon^2}{n} \Phi(\mathbf{w}^{(i,k)}) \right) + \left((1 + \epsilon\gamma) \mathbf{w}_{\bar{e}}^{(i,k)} + \frac{\epsilon^2\gamma}{n} \Phi(\mathbf{w}^{(i,k)}) \right) \\ &= \Phi(\mathbf{w}^{(i,k)}) + \epsilon \sum_{e \in H} \mathbf{r}_e^{(i,k)} + \epsilon\gamma \mathbf{r}_{\bar{e}}^{(i,k)} \\ &\leq \Phi(\mathbf{w}^{(i,k)}) + \frac{\epsilon}{\tau} \Psi(\mathbf{r}^{(i,k)}) + \epsilon\gamma \mathbf{r}_{\bar{e}}^{(i,k)}, \end{aligned}$$

where the last step follows from $\sum_{e \in H} \mathbf{r}_e^{(i,k)} \leq \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$ as guaranteed by Line 18 of Algorithm 10.

Now, if $\gamma = 1$, this means that $\mathbf{r}_{\bar{e}}^{(i,k)} \leq \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$, and when $\gamma \neq 1$, then $\gamma \mathbf{r}_{\bar{e}}^{(i,k)} = \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$. Therefore, combining both cases,

$$\begin{aligned} \Phi(\mathbf{w}^{(i,k+1)}) &\leq \Phi(\mathbf{w}^{(i,k)}) + \epsilon \tau^{-1} \Psi(\mathbf{r}^{(i,k)}) + \epsilon \tau^{-1} \Psi(\mathbf{r}^{(i,k)}) \\ &\leq \Phi(\mathbf{w}^{(i,k)}) \left(1 + \frac{2\epsilon e^\epsilon}{\tau} \right). \end{aligned}$$

Also note that for a width reduction step, $\Phi(\mathbf{w}^{(i,k+1)}) \geq \Phi(\mathbf{w}^{(i,k)})$. □

Change in Ψ

We now prove how our potential Ψ changes with a primal and width reduction step.

Lemma J.3. *If the parameters satisfy $\rho^2 \geq \tau \epsilon^{1/3} n^{-2\eta}$, $\rho \geq n^{1/2-3\eta}$, then after i primal and k width reduction steps, the potential $\Psi(\mathbf{r}^{(i,k)})$ satisfies,*

$$\Psi(\mathbf{r}^{(i,k)}) \geq \Psi(\mathbf{r}^{(0,0)})(1 - 10\epsilon\alpha\rho)^i \left(1 + \frac{\epsilon^{4/3}n^{-2\eta}}{10}\right)^k$$

Proof. We first bound the change in Ψ for a width reduction step. We will use Lemma A.3:

$$\Psi(\mathbf{r}') \geq \Psi(\mathbf{r}) + \sum_e \left(1 - \frac{\mathbf{r}_e}{\mathbf{r}'_e}\right) \mathbf{r}_e (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e^2, \text{ where } \Delta = \arg \min_{\Delta'} \sum_e \mathbf{r}_e (\tilde{\mathbf{C}}\Delta' - \tilde{\mathbf{d}})_e^2. \quad (39)$$

Width Steps. Suppose we have had i primal steps.

- We first consider the case when $\sum_{e \in S} \mathbf{r}_e^{(i,k)} \geq \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$, and in this case we perturb all edges in H and one extra edge $\bar{e} \in S \setminus H$. For edge \bar{e} , $\mathbf{w}_{\bar{e}}^{(i,k+1)} = \mathbf{w}_{\bar{e}}^{(i,k)}(1 + \epsilon\gamma) + \frac{\epsilon^2\gamma}{n} \Phi(\mathbf{w}^{(i,k)})$ and as a result,

$$\frac{\mathbf{r}_{\bar{e}}^{(i,k+1)} - \mathbf{r}_{\bar{e}}^{(i,k)}}{\mathbf{r}_{\bar{e}}^{(i,k+1)}} \geq \frac{\mathbf{w}_{\bar{e}}^{(i,k+1)} - \mathbf{w}_{\bar{e}}^{(i,k)}}{\mathbf{r}_{\bar{e}}^{(i,k+1)}} \geq \frac{\epsilon\gamma\mathbf{w}_{\bar{e}}^{(i,k)} + \frac{\epsilon^2\gamma}{n}\Phi(\mathbf{w}^{(i,k)})}{\mathbf{r}_{\bar{e}}^{(i,k+1)}} = \epsilon\gamma \frac{\mathbf{r}_{\bar{e}}^{(i,k)}}{\mathbf{r}_{\bar{e}}^{(i,k+1)}} \geq \frac{\epsilon\gamma}{1 + 2\epsilon},$$

where the last inequality follows from

$$\begin{aligned} \mathbf{r}_{\bar{e}}^{(i,k+1)} &= \mathbf{w}_{\bar{e}}^{(i,k)}(1 + \gamma\epsilon) + \frac{\epsilon^2\gamma}{2n}\Phi(\mathbf{w}^{(i,k)}) + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k+1)}) \\ &\leq \mathbf{w}_{\bar{e}}^{(i,k)} + \epsilon\mathbf{r}^{(i,k)} + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k)})(1 + 2\epsilon) \leq (1 + 3\epsilon)\mathbf{r}^{(i,k)}. \end{aligned}$$

Similarly for the other edges e in H ,

$$\frac{\mathbf{r}_e^{(i,k+1)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}} \geq \frac{\epsilon}{1 + 3\epsilon}.$$

We will use these bounds in Equation (39). Let us first consider the case when $\gamma = 1$.

$$\begin{aligned} \Psi(\mathbf{r}^{(i,k+1)}) &\geq \Psi(\mathbf{r}^{(i,k)}) + \sum_{e \in H \cup \{\bar{e}\}} \left(1 - \frac{\mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k+1)}}\right) \cdot \mathbf{r}_e^{(i,k)} \cdot (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \\ &\geq \Psi(\mathbf{r}^{(i,k)}) + \sum_{e \in H \cup \{\bar{e}\}} \frac{\epsilon}{(1 + 3\epsilon)} \mathbf{r}_e^{(i,k)} \rho^2 \\ &\geq \Psi(\mathbf{r}^{(i,k)}) + \frac{\epsilon\rho^2}{(1 + 3\epsilon)} \tau^{-1} \Psi(\mathbf{r}^{(i,k)}) \\ &= \Psi(\bar{\mathbf{r}}^{(i,k)}) \left(1 + \frac{\epsilon\rho^2}{(1 + 3\epsilon)\tau}\right), \end{aligned}$$

where the second step follows from $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \geq \rho$ for $e \in H$, and the third step follows from the definition of H that $H \subseteq S$ is maximal subset such that $\sum_{e \in H} \mathbf{r}_e^{(i,k)} \leq \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$, and so $\sum_{e \in H \cup \{\bar{e}\}} \mathbf{r}_e^{(i,k)} \geq \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$.

Now, in the case when $\gamma \neq 1$, we have $\gamma = \frac{\tau^{-1}}{\mathbf{r}_{\bar{e}}^{(i,k)}} \Psi(\mathbf{r}^{(i,k)}) < 1$,

$$\begin{aligned}
\Psi(\mathbf{r}^{(i,k+1)}) &\geq \Psi(\mathbf{r}^{(i,k)}) + \left(\frac{\epsilon\gamma}{1+3\epsilon} \right) \mathbf{r}_e^{(i,k)} \rho^2 \\
&\geq \Psi(\mathbf{r}^{(i,k)}) + \frac{\epsilon\rho^2}{(1+3\epsilon)\tau} \Psi(\mathbf{r}^{(i,k)}) \\
&\geq \Psi(\mathbf{r}^{(i,k)}) \left(1 + \frac{\epsilon\rho^2}{10\tau} \right).
\end{aligned}$$

Therefore, when $\sum_{e \in S} \mathbf{r}_e^{(i,k)} \geq \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$,

$$\Psi(\mathbf{r}^{(i,k+1)}) \geq \Psi(\mathbf{r}^{(i,k)}) \left(1 + \frac{\epsilon\rho^2}{10\tau} \right).$$

- Now, in the case when $\sum_{e \in S} \mathbf{r}_e^{(i,k)} < \tau^{-1} \Psi(\mathbf{r}^{(i,k)})$, i.e., $H = S$,

$$\sum_{e \notin H} \mathbf{r}_e^{(i,k)} |\mathbf{C}\Delta^{(i,k)} - \mathbf{d}|_e^3 \leq \max_{e \notin H} \{|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e\} \sum_{e \notin H} \mathbf{r}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \leq \rho \Psi(\mathbf{r}^{(i,k)}).$$

Since this is a width reduction step, we know that $\sum_e \mathbf{r}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq 2\rho \Psi(\mathbf{r}^{(i,k)})$, and therefore we must have,

$$\sum_{e \in H} \mathbf{r}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \geq \rho \Psi(\mathbf{r}^{(i,k)}).$$

Further, we can assume that for all $e \in H$, $|\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e \leq n^{1/2-\eta}\epsilon^{-1/3}$ since otherwise if there were one such edge, then using (39), $\mathbf{r} \geq \frac{\epsilon}{n} \Psi(\mathbf{r})$,

$$\Psi(\mathbf{r}^{(i,k+1)}) \geq \Psi(\mathbf{r}^{(i,k)}) + \frac{\epsilon^{1-2/3}}{(1+2\epsilon)} n^{1-2\eta} \frac{\epsilon}{n} \Psi(\mathbf{r}^{(i,k)}) \geq \Psi(\mathbf{r}^{(i,k)}) \left(1 + \frac{\epsilon^{4/3} n^{-2\eta}}{(1+\epsilon)} \right),$$

which gives us the required bound. Now,

$$\sum_{e \in H} \mathbf{r}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \geq \frac{\sum_{e \in H} \mathbf{r}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3}{\max_{e \in H} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e} \geq \frac{\epsilon^{1/3} \rho}{n^{1/2-\eta}} \Psi(\mathbf{r}^{(i,k)}).$$

Again, using this with (39) gives us,

$$\Psi(\mathbf{r}^{(i,k+1)}) \geq \Psi(\mathbf{r}^{(i,k)}) \left(1 + \frac{\epsilon^{4/3} \rho}{(1+3\epsilon)n^{1/2-\eta}} \right).$$

For the values of ρ and τ , such that $\rho^2/\tau \geq \epsilon^{1/3} n^{-2\eta}$, after every width reduction steps, we get,

$$\Psi(\mathbf{r}^{(i_k,k+1)}) \geq \Psi(\mathbf{r}^{(i_k,k)}) \left(1 + \frac{\epsilon^{4/3} n^{-2\eta}}{10} \right).$$

Primal Step. We next look at a primal step. For a primal step, $\mathbf{w}_e^{(i+1,k)} = \mathbf{w}_e^{(i,k)}(1 + \epsilon \vec{\alpha}(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}))$. Therefore,

$$\begin{aligned} \mathbf{r}_e^{(i+1,k)} &= \mathbf{w}_e^{(i+1,k)} + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i+1,k)}) \\ &\geq \mathbf{w}_e^{(i,k)} - \epsilon\alpha_+ \mathbf{w}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k)})(1 - \epsilon e^{\epsilon/2}\alpha_+) \\ &\geq \mathbf{w}_e^{(i,k)}(1 - \epsilon) + \frac{\epsilon}{2n}\Phi(\mathbf{w}^{(i,k)})(1 - \epsilon) = \mathbf{r}_e^{(i,k)}(1 - \epsilon). \end{aligned}$$

We also have,

$$\begin{aligned} \left| \frac{\mathbf{r}_e^{(i+1,k)} - \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i+1,k)}} \right| &\leq \frac{|\mathbf{w}_e^{(i+1,k)} - \mathbf{w}_e^{(i,k)}| + \frac{\epsilon}{2n}|\Phi(\mathbf{w}^{(i+1,k)}) - \Phi(\mathbf{w}^{(i,k)})|}{\mathbf{r}_e^{(i,k)}(1 - \epsilon)} \\ &\leq \frac{\epsilon\alpha_+ \mathbf{w}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e + \frac{\epsilon}{2n}e^\epsilon \epsilon\alpha_+ \Phi(\mathbf{w}^{(i,k)})}{\mathbf{r}_e^{(i,k)}(1 - \epsilon)} \\ &\leq \frac{\epsilon\alpha_+ \mathbf{r}_e^{(i,k)} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e + e^\epsilon \epsilon\alpha_+ \mathbf{r}_e^{(i,k)}}{\mathbf{r}_e^{(i,k)}(1 - \epsilon)} \\ &= e^\epsilon \epsilon\alpha_+ |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e + e^{2\epsilon} \epsilon\alpha_+. \end{aligned}$$

So we have,

$$\begin{aligned} \Psi(\mathbf{r}^{(i+1,k_i)}) &\geq \Psi(\mathbf{r}^{(i,k_i)}) - \sum_e \left| \frac{\mathbf{r}_e^{(i+1,k_i)} - \mathbf{r}_e^{(i,k_i)}}{\mathbf{r}_e^{(i+1,k_i)}} \right| \mathbf{r}_e^{(i,k_i)} (\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})_e^2 \\ &\geq \Psi(\mathbf{r}^{(i,k_i)}) - e^{2\epsilon} \epsilon\alpha \Psi(\mathbf{r}^{(i,k_i)}) - e^\epsilon \epsilon\alpha \sum_{e \in S_i} |\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}|_e^3 \mathbf{r}_e^{(i,k_i)} \\ &\geq \Psi(\mathbf{r}^{(i,k_i)}) - e^{2\epsilon} \epsilon\alpha \Psi(\mathbf{r}^{(i,k_i)}) - 2e^\epsilon \epsilon\alpha \rho \sum_{e \in S_i} \Psi(\mathbf{r}^{(i,k_i)}) \\ &\geq \Psi(\mathbf{r}^{(i,k_i)})(1 - 10\epsilon\alpha\rho). \end{aligned}$$

In the second last step we used the condition from Line 11. \square

We will now combine the changes in the two potentials similar to the proof of Theorem 2.1.

Proof of Theorem G.1

Proof. Let $\hat{\mathbf{x}} = \frac{\mathbf{x}}{T}$ be the solution returned by Algorithm 10. We would bound the objective value at $\hat{\mathbf{x}}$. Suppose the algorithm terminates in $T = \alpha^{-1}\epsilon^{-2} \ln n$ primal steps and $K \leq \tau/\epsilon^2$ width reduction steps. We can now apply Lemma J.2 to get,

$$\Phi(\mathbf{w}^{(T,K)}) \leq n \cdot e^{e^{\epsilon/2}\epsilon\alpha T} e^{2\epsilon e^\epsilon \tau K} \leq n^{O(\frac{1}{\epsilon})}.$$

We bound the ℓ_∞ norm of $\mathbf{C}\hat{\mathbf{x}} - \mathbf{d} = \frac{1}{T} \cdot \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})$ using the upper bound of the potential. Since $\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} \\ -\mathbf{C} \end{bmatrix}$, and $\tilde{\mathbf{d}} = \begin{bmatrix} \mathbf{d} \\ -\mathbf{d} \end{bmatrix}$, we have that the weights $\mathbf{w} \in \mathbb{R}^{2n}$.

Therefore, for $\mathbf{w}_+ \in \mathbb{R}^n$ and $\mathbf{w}_- \in \mathbb{R}^n$, we can write $\mathbf{w}^{(i,k)} = \begin{bmatrix} \mathbf{w}_+^{(i,k)} \\ \mathbf{w}_-^{(i,k)} \end{bmatrix}$, and we have that

$\Phi(\mathbf{w}) = \sum_{e \in [n]} \mathbf{w}_{+e} + \mathbf{w}_{-e}$. We can similarly define \mathbf{r}_+ and \mathbf{r}_- such that $\mathbf{r} = \begin{bmatrix} \mathbf{r}_+ \\ \mathbf{r}_- \end{bmatrix}$. Since $\Delta^{(i,k)}$ is obtained by solving,

$$\Delta^{(i,k)} = \arg \min_{\Delta} \sum_{e \in [2n]} \mathbf{r}_e^{(i,k)} (\tilde{\mathbf{C}}\Delta - \tilde{\mathbf{d}})_e^2 = \sum_{e \in [n]} (\mathbf{r}_+^{(i,k)} + \mathbf{r}_-^{(i,k)})_e (\mathbf{C}\Delta - \mathbf{d})_e^2,$$

the update rule $\mathbf{w}^{(i+1,k)} = \mathbf{w}^{(i,k)} \cdot (1 + \vec{\alpha}^{(i,k)}(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}}))$ implies that in every primal step,

$$(\mathbf{w}_+)_e^{(i+1,k)} = (\mathbf{w}_+)_e^{(i,k)} \cdot (1 + \vec{\alpha}^{(i,k)}(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})), \quad (\mathbf{w}_-)_e^{(i+1,k)} = (\mathbf{w}_-)_e^{(i,k)} \cdot (1 - \vec{\alpha}^{(i,k)}(\tilde{\mathbf{C}}\Delta^{(i,k)} - \tilde{\mathbf{d}})).$$

Now,

$$\begin{aligned} (\mathbf{w}_+)_e^{(T,K)} &= \mathbf{w}_e^{(0)} \cdot \prod_{i=0}^{T-1} \left(1 + \epsilon \vec{\alpha}_e^{(i,k_i)} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e\right) \\ &= \prod_{i: (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \geq 0} (1 + \epsilon \alpha_+ (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e) \cdot \prod_{i: (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e < 0} (1 + \epsilon \alpha_- (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e) \\ &\geq \exp\left(\epsilon(1 - \epsilon)\alpha \cdot \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e\right), \end{aligned}$$

where the second step follows from $\mathbf{w}^{(0)} = \mathbf{1}_n$, and $\vec{\alpha}_e^{(i,k_i)} = \alpha_+$ if $(\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \geq 0$ and $\vec{\alpha}_e^{(i,k_i)} = \alpha_-$ otherwise, the third step follows from $1 + \epsilon x \geq \exp(\epsilon(1 - \epsilon)x)$ for all $0 \leq x \leq 1$ and $1 + \epsilon x \geq \exp(\epsilon(1 + \epsilon)x)$ for all $-1 \leq x \leq 0$, and we have that $|\vec{\alpha}^{(i,k)} \cdot (\mathbf{C}\Delta^{(i,k)} - \mathbf{d})| \leq \frac{1}{10}$ by Lemma J.1. Similarly, we also get,

$$(\mathbf{w}_-)_e^{(T,K)} \geq \exp\left(\epsilon(1 - \epsilon)\alpha \cdot \sum_{i=0}^{T-1} -(\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e\right).$$

This implies that

$$\left| \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \right| \leq \frac{\ln((\mathbf{w}_+)_e^{(T,K)} + (\mathbf{w}_-)_e^{(T,K)})}{\epsilon(1 - \epsilon)\alpha} \leq \frac{\ln(\Phi(\mathbf{w}^{(T,K)}))}{\epsilon(1 - \epsilon)\alpha}.$$

So we have

$$\begin{aligned} \|\mathbf{C}\hat{\mathbf{x}} - \mathbf{d}\|_\infty &= \frac{1}{T} \max_e \left| \sum_{i=0}^{T-1} (\mathbf{C}\Delta^{(i,k_i)} - \mathbf{d})_e \right| \\ &\leq \frac{\ln(\Phi(\mathbf{w}^{(T,K)}))}{\alpha T} \\ &\leq \frac{\ln n + (1 + \epsilon)\epsilon\alpha T + (1 + \epsilon)}{\epsilon(1 - \epsilon)\alpha T} \\ &\leq 1 + 10\epsilon, \end{aligned}$$

We have shown that if the number of width reduction steps is bounded by K then our algorithm returns the required solution. We will next prove that we cannot have more than K width reduction steps.

We first show that if $\eta \leq 1/6$, the number of width steps K must be at most $\tilde{O}(1) \cdot T$. for the values of τ, ρ and α , the guarantee of Lemma J.3 becomes,

$$\begin{aligned}\Psi(\mathbf{r}^{(i,k)}) &\geq \Psi(\mathbf{r}^{(0,0)})(1 - 10\epsilon\alpha\rho)^T \left(1 + \frac{\epsilon^{4/3}n^{-2\eta}}{10}\right)^K \\ &= \exp\left\{\epsilon^{4/3}n^{-2\eta}K/20 - 20\epsilon\alpha\rho T\right\}\end{aligned}$$

Since $\Psi(\mathbf{r}) \leq (1 + \epsilon)\Phi \leq n^{O(1/\epsilon)}$, we must have,

$$n^{O(1/\epsilon)} \geq L \exp\left\{\epsilon^{4/3}n^{-2\eta}K/20 - 20\epsilon\alpha\rho T\right\},$$

or,

$$K \leq O(T) + \tilde{O}(1)n^{2\eta}/\epsilon^{7/3}.$$

Since $\eta \leq 1/6$, and $T = \alpha^{-1}\epsilon^{-2}\ln n$, $T + \tilde{O}(1)n^{2\eta}/\epsilon^{7/3} \leq \tau\epsilon^{-2}\ln n$ as required. Therefore the total number of iterations is at most,

$$T + K \leq \alpha^{-1}\epsilon^{-2}\log n + n^{2\eta}/\epsilon^{7/3} = \tilde{O}\left((n^{1/2-\eta} + n^{2\eta})/\epsilon^{7/3}\right).$$

□

Lower Bound on Ψ

Lemma J.4. For all i, k and $\mathbf{r}^{(i,k)}$ as defined in Algorithm 10, $\Psi(\mathbf{r}^{(i,k)}) \geq \frac{1}{1+2\epsilon} \cdot \Psi(\mathbf{r}^{(0,0)})$.

Proof. We first note that, $\Psi(\mathbf{r}^{(0,0)}) = 2(1+2\epsilon)\mathbf{d}^\top \left(\mathbf{I} - \mathbf{C}^\top (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}\right) \mathbf{d}$. Now, from Lemma A.3,

we know that for any $\mathbf{r}' \geq \mathbf{r}$, $\min_{\Delta} \sum_e \mathbf{r}'_e (\mathbf{C}\Delta - \mathbf{d})_e^2 \geq \min_{\Delta} \sum_e \mathbf{r}_e (\mathbf{C}\Delta - \mathbf{d})_e^2$. Let $\mathbf{r} = \begin{bmatrix} \mathbf{r}_+ \\ \mathbf{r}_- \end{bmatrix}$.

We also know that,

$$\Psi(\mathbf{r}^{(i,k)}) = \min_{\Delta} \sum_e \left(\mathbf{r}_+^{(i,k)} + \mathbf{r}_-^{(i,k)}\right)_e (\mathbf{C}\Delta - \mathbf{d})_e^2.$$

To prove our result, it is sufficient to prove that $\mathbf{r}_+^{(i,k)} + \mathbf{r}_-^{(i,k)} \geq 2$ because, then

$$\Psi(\mathbf{r}^{(i,k)}) \geq 2 \min_{\Delta} \sum_e (\mathbf{C}\Delta - \mathbf{d})_e^2 = 2\mathbf{d}^\top \left(\mathbf{I} - \mathbf{C}^\top (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}\right) \mathbf{d} = \frac{1}{1+2\epsilon} \Psi(\mathbf{r}^{(0,0)}).$$

Now,

$$\mathbf{r}_+^{(i,k)} + \mathbf{r}_-^{(i,k)} = \mathbf{w}_+^{(i,k)} + \mathbf{w}_-^{(i,k)} + \frac{2\epsilon}{n} \Phi(\mathbf{w}^{(i,k)}) \geq \mathbf{w}_+^{(i,k)} + \mathbf{w}_-^{(i,k)}.$$

Since the weights only increase during a width reduction step, we always have that,

$$\begin{aligned}(\mathbf{w}_+)^{(i,k)}_e &\geq \mathbf{w}_e^{(0,0)} \cdot \prod_{j=0}^{i-1} \left(1 + \epsilon \vec{\alpha}_e^{(j,k_j)} (\mathbf{C}\Delta^{(j,k_j)} - \mathbf{d})_e\right) \\ &= \prod_{j: (\mathbf{C}\Delta^{(j,k_j)} - \mathbf{d})_e \geq 0} (1 + \epsilon \alpha_+^{(j,k_j)} (\mathbf{C}\Delta^{(j,k_j)} - \mathbf{d})_e) \cdot \prod_{j: (\mathbf{C}\Delta^{(j,k_j)} - \mathbf{d})_e < 0} (1 + \epsilon \alpha_-^{(j,k_j)} (\mathbf{C}\Delta^{(j,k_j)} - \mathbf{d})_e) \\ &\geq \exp\left(\epsilon(1 - \epsilon)\alpha \cdot \sum_{j=0}^{i-1} (\mathbf{C}\Delta^{(j,k_j)} - \mathbf{d})_e\right),\end{aligned}$$

and similarly,

$$(\mathbf{w}_-)_e^{(i,k)} \geq \exp\left(-\epsilon(1-\epsilon)\alpha \cdot \sum_{j=0}^{i-1} (\mathbf{C}\Delta^{(j,k_j)} - \mathbf{d})_e\right).$$

The function $e^x + e^{-x} \geq 2$ for all $x \in \mathbb{R}$, since it is minimized at $x = 0$. Therefore, for all e

$$\left(\mathbf{r}_+^{(i,k)} + \mathbf{r}_-^{(i,k)}\right)_e \geq \left(\mathbf{w}_+^{(i,k)} + \mathbf{w}_-^{(i,k)}\right)_e \geq 2,$$

concluding the proof of our result. □