Distributed Load Balancing: A New Framework and Improved Guarantees

Sara Ahmadian
Google Research, New York, NY, USA
sahmadian@google.com

Allen Liu
MIT, Cambridge, MA, USA
cliu568@mit.edu

Binghui Peng
Columbia University, New York, NY, USA
bp2601@columbia.edu

Morteza Zadimoghaddam
Google Research, Cambridge, MA, USA
zadim@google.com

Abstract

Inspired by applications on search engines and web servers, we consider a load balancing problem with a general *convex* objective function. In this problem, we are given a bipartite graph on a set of sources S and a set of workers W and the goal is to distribute the load from each source among its neighboring workers such that the total load of workers are as balanced as possible. We present a new distributed algorithm that works with *any* symmetric non-decreasing convex function for evaluating the balancedness of the workers' load. Our algorithm computes a nearly optimal allocation of loads in $O(\log n \log^2 d/\epsilon^3)$ rounds where n is the number of nodes, d is the maximum degree, and ϵ is the desired precision. If the objective is to minimize the maximum load, we modify the algorithm to obtain a nearly optimal solution in $O(\log n \log d/\epsilon^2)$ rounds. This improves a line of algorithms that require a polynomial number of rounds in n and d and appear to encounter a fundamental barrier that prevents them from obtaining poly-logarithmic runtime [6, 7, 13, 15]. In our paper, we introduce a novel primal-dual approach with multiplicative weight updates that allows us to circumvent this barrier. Our algorithm is inspired by [1] and other distributed algorithms for optimizing linear objectives but introduces several new twists to deal with general convex objectives.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Load balancing, Distributed algorithms

Digital Object Identifier 10.4230/LIPIcs.ITCS.2021.79

Funding *Binghui Peng*: Research supported by NSF IIS-1838154, NSF CCF-1703925 and NSF CCF-1763970.

1 Introduction

Emerging web based services including commercial web search engines face challenging resource efficiency targets in their serving data centers. Motivated by the growing demand in fast responding services, they aim for sub-second latency targets. Therefore they replicate the data across distributed machines in data centers to allow for serving queries in parallel as well as cloning the search algorithm to expedite computation tasks. With billions of queries to serve on a daily basis [14], load balancing becomes a critical challenge in resource efficiency and optimizing the computation fleet.

From the combinatorial optimization perspective, one can formulate the serving requirements with packing and covering constraints and model this problem as an allocation/matching instance in a bipartite graph. Although feasibility of the allocation is the first problem to study, in practice, we face a wider range of objectives to optimize. Emergency mechanisms in data centers allow the excess load to be served with the buffer capacities locally or be shifted to alternative data centers globally with services like Global Server Load Balancing.

This motivates service level objectives (SLO) in terms of quantile statistics of machine utilization values or other convex functions that are much more sensitive to higher utilization values instead of standard linear objective functions in matching theory that have uniform partial derivatives across the whole range of valid utilization values. One particular frequently occurring scenario is when the underlying properties of the load balancing instance determines some phase transition utilization threshold (say 0.95) beyond which the service starts to deteriorate. Thus, the cost objective function we are trying to minimize should have completely different behaviours on the two sides of this threshold and linear functions are unable to capture this exponential growth in the cost function.

To accommodate this wide range of objectives, in this paper we focus on the load balancing problem with a general *convex* objective function. The problem is defined on a bipartite graph, with a set of sources S on one side, and a set of workers W on the other. For each source, we must distribute its load among its neighboring workers. The goal is to minimize a convex function of the workers' loads (where the load of a worker is the total load it receives from all incoming sources). Fractional allocations are allowed, as in the real-world setting, there are usually huge amounts of query requests coming from each source.

The problem described above can be solved as a convex program. However, in real-world settings, the graph could contain billions of nodes and be too large to store in one piece. Thus, for an algorithm to be scalable, it has to be implementable in a *distributed* manner. In this paper, we work in the CONGEST model that is standard in distributed algorithms literature (see e.g. [4]). Under this model, computation proceeds in rounds and in each round, each node may send a logarithmic number of bits to each of its neighbors. Previous distributed algorithms for load balancing (with a convex objective), such as [6, 7, 13, 15], require a number of distributed rounds that is *polynomial* in the number of nodes or the maximum degree of a node. This could still be prohibitive for real-world applications where each node could have a large number of neighbors. The main contribution of this paper is to provide *the first distributed algorithm* that computes an approximately optimal solution to the load balancing problem with a convex objective and runs in a *poly-logarithmic* number of distributed rounds. Our main theorem is stated below.

▶ **Theorem 1** (Informal). *Assume the objective function Φ is convex, symmetric, and non-decreasing in each variable. Then for any $\epsilon < 1$, our algorithm computes a $(1+\epsilon)$-approximation to the optimal solution in $O\left(\frac{\log n \log^2 d}{\epsilon^3}\right)$ rounds, where n, d denote the number of nodes and the maximum degree respectively.* [1]

In the special case where the objective is the max function, i.e. the goal is to minimize the maximum load, we obtain a distributed algorithm with an improved round complexity of $O\left(\frac{\log n \log d}{\epsilon^2}\right)$, which is faster than a direct application of a parallel mixed positive LP solver (see [11]) by an $O\left(\frac{\log n}{\epsilon}\right)$ factor.

[1] Technically, we need a few additional assumptions on the objective function Φ in order to obtain a $(1+\epsilon)$-approximation in terms of objective value. Formally, our theorem is stated in terms of an ϵ-approximate solution which is defined in the main body but for natural objective functions such as L^p norms for $p > 1$, our algorithm obtains a $(1+\epsilon)$-approximation in objective value. See the remark at the end of Section 2 for more details.

▶ **Theorem 2** (Informal). *When the objective is the max function, our algorithm computes a $(1+\epsilon)$-approximation to the optimal solution and runs in $O\left(\frac{\log n \log d}{\epsilon^2}\right)$ rounds.*

1.1 Related work

There is extensive work on distributed algorithms for optimizing linear objective functions such as packing, covering and positive LPs [2,3,5,10,11,17,18]. In these settings, distributed algorithms with poly-logarithmic convergence rates are known. In particular, for general positive LPs, there is a distributed algorithm that computes a $(1+\epsilon)$-approximation in $O(\frac{\log^3 n}{\epsilon^3})$ rounds; for the special case of pure packing and pure covering LPs, a better runtime of $O\left(\frac{\log^2 n}{\epsilon^2}\right)$ is known [11]. We refer the interested reader to the thesis [16] for a more detailed survey. While the techniques in these works can be applied to our problem when the objective function is linear, optimizing a general convex objective is significantly different. Whereas for a linear objective, the optimum always occurs at one of the vertices of the feasible polytope, for a convex objective, the optimum may be in the interior. Therefore, it seems that existing algorithms for optimizing linear objectives cannot be directly applied.

As mentioned previously, there are works that study a convex load balancing objective [6,7,13,15]. The algorithms proposed in these works have distributed runtime that depends polynomially on the number of nodes or the maximum degree whereas our algorithm has only polylogarithmic dependence on these parameters.

There has also been work on discrete load balancing i.e. when the sources are not divisible (so fractional allocations are not allowed). A recent line of work [4,8,9] obtains constant-factor approximation algorithms in the local and congest models. More specifically, Czygrinow et al. [8] give a distributed 2-approximation algorithm that runs in $O(d^5)$ rounds. Assadi et al. [4] gives an $O(1)$-approximation algorithm for unweighted loads and an $O(\log n)$ approximation algorithm for weighted loads with polylogarithmic round complexity in the congest model. For the less restrictive local model, they present an $O(1)$-approximation algorithm for weighted loads with polylogarithmic round complexity. In general, when the sources are not divisible, one cannot hope to compute, say, a $(1+\epsilon)$-approximation efficiently. Thus, these works focus on constant-factor approximation, whereas in our setting, we focus on computing a nearly optimal solution.

1.2 Technical Overview

One key limitation for previous works on (continuous) load balancing [6,7,13,15] is that they rely on an algorithm that, for each source, additively shifts load from higher-loaded neighbors to lower-loaded neighbors. However, the step size must be set to $O(1/d)$ in order for the algorithm to be stable and thus the number of rounds required depends linearly (or even worse) on the maximum degree d.

Since the load balancing problem is convex, another natural approach is to apply general convex optimization algorithms that can be implemented in a distributed manner. However, straight-forward applications of first-order convex optimization algorithms also get stuck with a linear dependence on d because both the diameter of the feasible polytope and the condition number of the convex optimization problem can depend linearly on the maximum degree.

We circumvent the aforementioned limitations by adopting a different algorithmic framework and we introduce a novel algorithm based on a primal-dual approach with multiplicative-weight updates. One of the central insights in our algorithm is that if we know the target capacities in the optimal solution, then we can compute the allocation of the sources that achieves the optimum as this essentially reduces to solving a linear problem. More generally, for a given set of target capacities, we can essentially test whether it is achievable. This motivates the following iterative procedure: we start with very high target capacities and iteratively update the target capacities downwards while checking feasibility until we reach a solution that is barely feasible.

For checking feasibility, we use a proportional allocation algorithm based on the work in [1] for maximum matching. We update the load assignment according to the proportional allocation algorithm and then update the target capacities based on whether each worker has too much or too little load. We start our algorithm with a feasible solution (by setting the target capacities to be very high) and can easily maintain feasibility throughout the entire process. The difficulty lies in proving optimality. An important observation is that after running the proportional allocation algorithm for a sufficient number of iterations, if there are workers whose load differs significantly from their target capacity, then it is possible to (implicitly) construct a certificate that this imbalance must happen in any feasible allocation. For the special case when the objective is the max function, it is not too difficult to complete the proof using the above ideas as it suffices to maintain the same target capacity for all the workers and decrease all capacities together. When the algorithm stops, we only need to show that there exists a set of workers that is *saturated*, in the sense that their total capacity is roughly equal to the total load of the sources whose neighbors are all included in this set. For general convex objectives, we will need a more refined analysis since the workers may have different target capacities. We will prove that the solution that we compute is optimal at multiple levels. In particular, our algorithm allows us to (implicitly) construct a multi-level cut that serves as a hierarchy of certificates that, when combined, imply that the entire solution is nearly optimal.

2 Preliminary

We now formalize our problem and introduce notation. In the load balancing problem, the input is a bipartite graph $G(S, W, E)$, where we use S ($|S| = n_S$) to denote the set of sources and W ($|W| = n_W$) to denote the set of workers. We write $n = n_S + n_W$. We assume there is a load associated with each source $s \in S$. For simplicity of presentation, we assume each source has one unit of load, although following the same proof method, the results obtained in this paper can be generalized to arbitrary weighted loads. For each source $s \in S$, we use N_s to denote the set of workers that are connected to the source s. Similarly, we use N_w to denote all the sources that are connected to worker w. We use d to denote the maximum degree of a node (source or worker) in the graph. For a subset of workers $X \subset W$, let $N(X) \subset S$ be the set of sources s with the property that all of the neighbors of s are in X.

We want to assign loads along the edges of the graph such that all of the sources are served and the loads of the workers are as balanced as possible. We use $x_{s,w}$ to denote the amount of load assigned from source s to worker w. We assume the load is splittable and fractional allocations are allowed. For each worker w, define its load L_w to be

$$L_w = \sum_{s \in N_w} x_{s,w}.$$

In this paper, our objective is to minimize some convex function of the loads. We assume the objective function $\Phi : \mathbb{R}^{n_W} \to \mathbb{R}$ is symmetric, convex, and non-decreasing in each variable. Formally, we aim to solve the following optimization problem in a distributed manner:

$$\begin{aligned} \text{minimize} \quad & \Phi(L_1, \ldots, L_{n_W}) \\ \text{subject to} \quad & \sum_{w \in N_s} x_{s,w} = 1 \ \forall s \in S \\ & L_w = \sum_{s \in N_w} x_{s,w} \ \forall w \in W \\ & x_{s,w} \geq 0 \ \forall s \in S, w \in W \\ & x_{s,w} = 0 \ \forall (s,w) \notin E \end{aligned} \tag{1}$$

Given any load vector $L = (L_1, \cdots, L_{n_W}) \in \mathbb{R}^{n_W}$, we say it is *feasible* if there exists a feasible assignment $\{x_{s,w}\}_{s \in S, w \in W}$ that satisfies the constraints of Eq. (1). Our algorithm will find an ϵ-approximate solution as defined below.

▶ **Definition 3** (ϵ-approximate solution). *We say $L = (L_1, \cdots, L_{n_W})$ is an ϵ-approximate solution, if L is feasible and for any other feasible solution $(L_1', \cdots, L_{n_W}')$, we have*

$$\Phi(L_1, \ldots, L_{n_W}) \leq \Phi\left((1+\epsilon)L_1', \ldots, (1+\epsilon)L_{n_W}'\right).$$

We are primarily interested in the case where $1/\epsilon$ is constant or poly-logarithmic in n so runtime that is polynomial in $1/\epsilon$ is acceptable.

▶ Remark 4. The definition of ϵ-approximate solution is stated in terms of scaling the *inputs* of the convex function rather than scaling the *value* of the convex function itself. An ϵ-approximate solution would directly imply a $(1+O(\epsilon))$ approximation on the optimal value for any Lipschitz continuous function together with a mild lower bound assumption on the optimal value. In particular, it implies $(1+\epsilon)$-approximation (in terms of objective value) for widely used functions such as L_p norms for $p \geq 1$.

In general, it is impossible to achieve a multiplicative approximation without additional assumptions on the objective. For instance, if we take $\Phi = \max\left(\max(L_1, \ldots, L_{n_W}) - \mathsf{OPT}, 0\right)$ where OPT is the minimum possible value of $\max(L_1, \ldots, L_{n_W})$, then achieving a multiplicative approximation would require solving the problem exactly.

3 Algorithm

We first give a high level overview of our main algorithm (the pseudocode is given in Algorithm 1). The algorithm maintains a target capacity C_w and a weight a_w for each worker $w \in W$. For any source $s \in S$, the amount of load it assigns to its neighbor worker w ($w \in N_w$) is proportional to the weight a_w (see Line 13). Our algorithm always maintains a (near) feasible solution under the target capacity. It gradually decreases the target capacity and adjusts the weights, until reaching a nearly optimal solution. The algorithm updates the capacity once per epoch (see Line 8 to Line 12). While in each epoch, the algorithm freezes the target capacity and attempts to find a feasible assignment via the Proportional Allocation algorithm (see Algorithm 2). The Proportional Allocation algorithm was originally proposed in [1] for finding an approximate maximum matching. It adjusts the weight of workers based on the following rules: If the current load exceeds the target capacity, it decreases the weight by a multiplicative factor of $(1+\epsilon)$; it increases the weight otherwise. After repeating the weight updating rules for $\tilde{O}\left(\frac{\log n \log d}{\epsilon^2}\right)$ steps, our algorithm updates the target capacities on workers based on the gap between the target capacity and the current load. The algorithm will *fix* workers whose load significantly exceeds the target capacity meaning that for these workers, the target capacity never changes anymore.

Algorithm 1 GENERAL LOAD BALANCING.

```
1: **Input**: Graph $G$, set of workers $W$, set of sources $S$
2: Initialize weights $a_w = 1$ for all workers $w \in W$.
3: Initialize capacity upper bounds $C_w = d$ for all workers $w \in W$
4: Set all workers $w$ to be *unfixed*
5: Set $A = \frac{2\log(d/\epsilon)}{\epsilon}$ and $B = \frac{100\log(n/\epsilon)\cdot\log(d/\epsilon)}{\epsilon^2}$
6: for $r = 0, 1, \ldots, A-1$ do
7:     $x \leftarrow$ PROPORTIONAL ALLOCATION$(G, C, a, \epsilon, B)$
8:     for $w \in W$ do
9:         if $w$ is *unfixed* and $L_w < (1+10\epsilon)C_w$ then
10:            Set $C_w \leftarrow \frac{C_w}{1+\epsilon}$
11:        else
12:            Set $w$ to *fixed*
13: return $x_{s,w} = \frac{a_w}{\sum_{w \in N_s} a_w}$
```

The pseudocode of the PROPORTIONAL ALLOCATION algorithm is given in Algorithm 2. We note that each time we run PROPORTIONAL ALLOCATION and update the weights a_w, we *do not* reinitialize the weights. We continue updating from the weights computed in the previous step.

Algorithm 2 PROPORTIONAL ALLOCATION [1] (G, C, a, ϵ, B).

```
Input: Graph $G$, set of workers $W$, set of sources $S$
Input: Target capacity $C_w$ for each worker
Input: Initial weight $a_w$ for each worker
Input: Precision $\epsilon$
input: Number of rounds $B$
for $t = 0, 1, \ldots, B-1$ do
    Set $x_{s,w} = \frac{a_w}{\sum_{w \in N_s} a_w}$ for all edge variables $x_{s,w}$
    for $w \in W$ do
        If $L_w > C_w$ then update $a_w \leftarrow \frac{a_w}{1+\epsilon}$
        If $L_w < C_w$ then update $a_w \leftarrow (1+\epsilon)a_w$
return $x_{s,w} = \frac{a_w}{\sum_{w \in N_s} a_w}$
```

4 Analysis

Our main result is formally stated in Theorem 5, we sketch the high-level idea of the proof here. The main idea in the proof of Theorem 5 is to show that if we have fixed the sets of workers $W_1, W_2, \ldots, W_r$ in iterations $1, 2, \ldots, r$, then the number of sources whose only neighbors are in the set $W_1 \cup \cdots \cup W_r$ is at least $(1 - O(\epsilon))\sum_{w \in W_1 \cup \cdots \cup W_r} C_w$. This would then certify that our solution is essentially optimal for the total load among the set of workers in $W_1 \cup \cdots \cup W_r$. While this claim is not technically true, Lemma 12 is a slight modification that involves considering a superset of $W_1 \cup \cdots \cup W_r$ and it suffices for our purposes. Once we have a hierarchy of certificates for $r = 1, 2, \ldots, A-1$ we can prove that the solution computed by our algorithm is essentially optimal overall.

▶ **Theorem 5.** *Assume the objective function $\Phi : \mathbb{R}^{nw} \to \mathbb{R}^+$ is convex, symmetric, and non-decreasing in each variable. For $0 < \epsilon < 1$, Algorithm 1 computes an $O(\epsilon)$-approximate solution in $O\left(\frac{\log n \log^2 d}{\epsilon^3}\right)$ distributed rounds.*

In the proof of Theorem 5, the bulk of the work is in proving Lemma 12. We will first prove some basic facts about the behavior of the loads and target capacities throughout the execution of Algorithm 1 in Section 4.1. We then introduce the concept of majorization for analyzing convex functions in Section 4.2. In Section 4.3, we prove Lemma 12. A key observation about the proportional allocation algorithm of [1] is that at the end, if the load on a worker is significantly less than the target capacity, its weight a_w must be increased at every round and if the load on a worker is significantly more than the target capacity, its weight a_w must be decreased at every round. Thus, there must be a large multiplicative gap between weights on underallocated and overallocated workers. Since loads are allocated proportionally, if some source is connected to both underallocated workers and overallocated workers, almost all of the load is actually being sent to the underallocated workers. Exploiting this intuition (we will need a slightly more precise statement in the proof), we can construct the desired certificate and complete the proof of Lemma 12. Finally, combining Lemma 12 with the tools introduced in Section 4.2 for analyzing convex functions, we complete the proof of Theorem 5.

4.1 Basic Observations

Notation

For a load variable L_w and indices $0 \le r < A, 0 \le t \le B$, we let $L_w^{r,t}$ denote its value when the algorithm is executed to the timestep r, t i.e. we have completed r full iterations of Proportional Allocation and t rounds within the next iteration of Proportional Allocation. We adopt the same notation for $x_{s,w}$, s_w and C_w.
Below is an informal summary of the properties that we will prove in this section.

- Loads gradually move toward the target capacities
- Loads never significantly exceed the target capacity
- For fixed workers, their load is roughly equal to their target capacity
- Any significantly underallocated workers must have their weight increased at every previous timestep

We begin with a simple observation that after each weight update, the load on each worker moves toward the target capacity.

▶ **Lemma 6.** *Consider indices $0 \le r < A, 0 \le t < B$ then*
- *If $L_w^{r,t} > C_w^{r,t}$ then $\frac{L_w^{r,t}}{(1+\epsilon)^2} \le L_w^{r,t+1} \le L_w^{r,t}$*
- *If $L_w^{r,t} < C_w^{r,t}$ then $L_w^{r,t} \le L_w^{r,t+1} \le (1+\epsilon)^2 L_w^{r,t}$*

Proof. We prove the the first claim and the second one follows from the same argument. Suppose $L_w^{r,t} > C_w^{r,t}$, then we know that $a_w^{r,t+1} = a_w^{r,t}/(1+\epsilon)$. Together with the fact that $a_{w'}^{r,t} \in [(1+\epsilon)^{-1} a_{w'}^{r,t}, (1+\epsilon) a_{w'}^{r,t}]$ holds for all worker $w' \in W$, we have

$$L_w^{r,t+1} = \sum_{s \in N_w} \frac{a_w^{r,t+1}}{\sum_{w' \in N_s} a_{w'}^{r,t+1}} \le \sum_{s \in N_w} \frac{a_w^{r,t} \cdot (1+\epsilon)^{-1}}{\sum_{w' \in N_s} a_{w'}^{r,t} \cdot (1+\epsilon)^{-1}} = L_w^{r,t}$$

and

$$L_w^{r,t+1} = \sum_{s \in N_w} \frac{a_w^{r,t+1}}{\sum_{w' \in N_s} a_{w'}^{r,t+1}} \ge \sum_{s \in N_w} \frac{a_w^{r,t} \cdot (1+\epsilon)^{-1}}{\sum_{w' \in N_s} a_{w'}^{r,t} \cdot (1+\epsilon)} = \frac{L_w^{r,t}}{(1+\epsilon)^2}.$$ ◀

Next, we observe that the load on any worker can never significantly exceed its target capacity.

▶ **Lemma 7.** *For all workers w, we have at all timesteps r, t,*

$$L_w^{r,t} \leq (1+10\epsilon)(1+\epsilon)C_w^{r,t}.$$

Proof. We prove the claim by induction on r and t. The base case is clearly true as

$$L_w^{0,0} \leq d = C_w^{0,0}.$$

Suppose the claim holds up to r, t. If $0 \leq t < B$, then Lemma 6 implies the desired for $r, t+1$. If $t = B$, then we are in one of the following two cases

- $L_w^{r,B} < (1+10\epsilon)C_w^{r,B}$, which implies $L_w^{r+1,0} < (1+10\epsilon)(1+\epsilon)C_w^{r+1,0}$
- $L_w^{r,B} \geq (1+10\epsilon)C_w^{r,B}$, which implies $C_w^{r+1,0} = C_w^{r,B}$

The first case is clearly resolved. For the second case, we can use the induction hypothesis to get the desired. ◀

When a worker becomes *fixed*, we observe that at all future timesteps, its load is close to its target capacity.

▶ **Lemma 8.** *For a worker w, once w is fixed, we have*

$$L_w^{r,t} \geq \frac{C_w^{r,t}}{(1+\epsilon)^2}$$

holds for all ***future*** *timesteps.*

Proof. When w is *fixed*, we must have $L_w > C_w$. Combining Lemma 6 with the fact that we no longer update C_w, we get the desired. ◀

For any worker whose load is significantly lower than its capacity, we note that its weight a_w must have been increased at **every** previous timestep.

▶ **Lemma 9.** *Consider a worker w such that for some $0 \leq r < A$, if*

$$L_w^{r,B} \leq \frac{C_w^{r,B}}{(1+\epsilon)^2},$$

then

$$a_w = (1+\epsilon)^{(r+1)B},$$

i.e. if a worker is significantly underallocated when we reach the capacity update step, then its weight must have been increased at every step.

Proof. This follows immediately from Lemma 6 and the fact that the capacities C_w are weakly decreasing. ◀

4.2 Majorization

We present a basic inequality about convex functions that will be useful later on for bounding the objective value. We first introduce the concept of *majorization*.

▶ **Definition 10** (Majorization). *For two sequences of real numbers* $(x_1, \ldots, x_n)$ *and* $(y_1, \ldots, y_n)$*, let* π, σ *be permutations such that*

$$x_{\pi(1)} \geq \cdots \geq x_{\pi(n)}, \; y_{\sigma(1)} \geq \cdots \geq y_{\sigma(n)}.$$

We say $(x_1, \ldots, x_n)$ ***weakly majorizes*** $(y_1, \ldots, y_n)$ *if for all* $1 \leq k \leq n$

$$\sum_{i=1}^{k} x_{\pi(i)} \geq \sum_{i=1}^{k} y_{\sigma(i)}.$$

If we also have that

$$\sum_{i=1}^{n} x_{\pi(i)} = \sum_{i=1}^{n} y_{\sigma(i)}$$

then we say $(x_1, \ldots, x_n)$ ***majorizes*** $(y_1, \ldots, y_n)$

Intuitively, a sequence $(x_1, \ldots, x_n)$ majorizes a sequence $(y_1, \ldots, y_n)$ if the terms of $(x_1, \ldots, x_n)$ are more imbalanced. The following inequality states that a symmetric convex function takes larger values when the inputs are more imbalanced:

▶ **Lemma 11.** *Let* f *be a convex function that is symmetric and non-decreasing in each of the variables. Given sequences* $(x_1, \ldots, x_n)$ *and* $(y_1, \ldots, y_n)$ *such that* $(x_1, \ldots, x_n)$ *weakly majorizes* $(y_1, \ldots, y_n)$*, we have*

$$f(x_1, \ldots, x_n) \geq f(y_1, \ldots, y_n).$$

If $(x_1, \ldots, x_n)$ *majorizes* $(y_1, \ldots, y_n)$ *then the above inequality holds without the assumption that* f *is non-decreasing.*

Proof. [12] proves the above inequality when $(x_1, \ldots, x_n)$ majorizes $(y_1, \ldots, y_n)$, we adapt it to the case that $(x_1, \ldots, x_n)$ weakly majorizes $(y_1, \ldots, y_n)$. In particular, we prove that there exists a sequence $(y'_1, \ldots, y'_n)$ such that $y'_i \geq y_i$ for all $1 \leq i \leq n$ and $(x_1, \ldots, x_n)$ majorizes $(y'_1, \ldots, y'_n)$. WLOG, we assume $x_1 \geq \cdots \geq x_n$ and $y_1 \geq \cdots \geq y_n$. We construct the sequence as follows. We first set y'_1 to be the largest value so that $(x_1, \ldots, x_n)$ weakly majorizes $(y'_1, y_2 \ldots, y_n)$. After determining the value of y'_1, we set y'_2 to be the largest value so that $(x_1, \ldots, x_n)$ weakly majorizes $(y'_1, y'_2, y_3, \ldots, y_n)$. We repeat this process to set all of $y'_3, \ldots, y'_n$. Now, for each index $i \in [n]$, there exists an index j with $j \geq i$ such that

$$x_1 + \cdots + x_j = y'_1 + \cdots + y'_i + y_{i+1} + \cdots + y_j,$$

as otherwise, this would contradict the maximality of y'_i. Hence, we conclude that

$$x_1 + \cdots + x_n = y'_1 + \cdots + y'_n,$$

so $(x_1, \ldots, x_n)$ majorizes $(y'_1, \ldots, y'_n)$, as desired. ◀

4.3 Main Proof

Now we are ready to analyze the performance of the algorithm. The following lemma is essential to our proof. Intuitively, it allows us to construct a set of cuts that “certify” that the solution computed by the algorithm is essentially optimal.

▶ **Lemma 12.** *Let $X_0 = \emptyset$. We can construct sets $X_1, \ldots, X_A \subset W$ with the following properties:*

- $X_1 \subset X_2 \subset \cdots \subset X_A$
- $|N(X_r)| \geq \sum_{i=0}^{r-1} |X_{i+1} \backslash X_i| \cdot \frac{d}{(1+\epsilon)^{i+10}}$
- *X_i contains all of the workers that are fixed after executing* PROPORTIONAL ALLOCATION *and capacity updates for $r = 0, 1, \ldots, i-1$*

Proof. We prove the claim by induction. Assume that we have already constructed $X_1, \ldots, X_{i-1}$. Let F^i be the set of workers that become *fixed* after completing the capacity updates for $r = i-1$ but are not fixed before this. For any worker $w \in F^i$, we must have

$$a_w^{i-1,t+1} = \frac{a_w^{i-1,t}}{1+\epsilon} \quad \forall\, 0 \leq t < B.$$

This is because if this was not the case, we must have $L_w^{i-1,t_0} < C_w^{i-1,t_0}$ for some t_0 and by Lemma 6, this implies $L_w^{i-1,t} \leq (1+\epsilon)^2 C_w^{i-1,t}$ holds for all $t \geq t_0$. This contradicts the fact that worker w gets fixed in the execution of the capacity updates for $r = i-1$. Thus, we conclude for any $w \in F^i$

$$a_w^{i-1,B} \leq (1+\epsilon)^{(i-1)B}.$$

Let $Y = F^i \backslash (F^i \cap X_{i-1})$. If Y is empty then it suffices to set $X_i = X_{i-1}$. Assume Y is not empty, then for any integer $0 \leq j \leq B$, define Z_j as

$$Z_j := \{w | w \in W, a_w \leq (1+\epsilon)^{(i-1)B+j}\}.$$

We note that $Y \subseteq Z_0 \subseteq Z_1 \subseteq \cdots \subseteq Z_B$ and therefore $Z_0 \neq \emptyset$.

Since $B \geq \frac{100 \log(n/\epsilon) \cdot \log(d/\epsilon)}{\epsilon^2}$, there must exist some index j satisfying

$$10 \log\left(\frac{d}{\epsilon}\right) \cdot \frac{1}{\epsilon} \leq j < B$$

and

$$|Z_j \backslash X_{i-1}| \leq (1+\epsilon) \cdot \left| Z_{j-10\log(\frac{d}{\epsilon})\cdot\frac{1}{\epsilon}} \backslash X_{i-1} \right|. \tag{2}$$

We set $X_i = Z_j \cup X_{i-1}$ and prove it satisfies all three properties in the rest of the proof. The first and third properties are clearly satisfied, i.e., $Z_j \cup X_{j-1}$ is a superset of X_{k-1} and it contains all of the workers that are fixed after i executions of PROPORTIONAL ALLOCATION. It remains to verify the second one

For any worker $w \in Z_{j-10\log(\frac{d_{\max}}{\epsilon})\cdot\frac{1}{\epsilon}}$, if the worker w is connected to some source s, such that s has another neighbor worker w' with $w' \notin Z_j$, then we have

$$x_{s,w}^{i-1,B} \leq \frac{a_w^{i-1,B}}{a_w^{i-1,B} + a_{w'}^{i-1,B}} < \frac{(1+\epsilon)^{(i-1)B+j-10\log(\frac{d}{\epsilon})\cdot\frac{1}{\epsilon}}}{(1+\epsilon)^{(i-1)B+j}}. \leq \left(\frac{\epsilon}{d}\right)^{10}$$

Therefore, the total contributions $x_{s,w}^{i-1,B}$ from all sources s with a neighbor not in Z_j to the worker w is at most

$$\sum_{\substack{s \in N_w \\ N_s \cap (W \backslash Z_j) \neq \emptyset}} x_{s,w}^{i-1,B} \leq \left(\frac{\epsilon}{d}\right)^{10} \cdot d \leq \left(\frac{\epsilon}{d}\right)^{9}. \tag{3}$$

Now we bound the changes of $|N(X_i)|$ comparing to $|N(X_{i-1})|$. Consider the graph $G^{i-1,B}$ at timestep $(i-1,B)$. We restrict the graph to the one induced by the set of workers Z_j and the set of sources $N(Z_j)$. For each worker $w \in Z_{j-10\log(\frac{d}{\epsilon})\cdot\frac{1}{\epsilon}} \backslash X_{i-1}$, its load in $G^{i-1,B}$ satisfies

$$L_w^{i-1,B} \geq \frac{C_w^{i-1,B}}{(1+\epsilon)^2} \geq \frac{d}{(1+\epsilon)^{i+1}} \tag{4}$$

The first step comes from Lemma 9, the second step comes from the fact that $C_w^{i-1,B} = d/(1+\epsilon)^{i-1}$.

Thus the load of worker w in the restricted graph is at least

$$\begin{aligned}\sum_{\substack{s\in N_w\\ N_s\cap(W\backslash Z_j)=\emptyset}} x_{s,w}^{r,A_r-1,B_r} &= L_w^{r,A_r-1,B_r} - \sum_{\substack{s\in N_w\\ N_s\cap(W\backslash Z_j)\neq\emptyset}} x_{s,w}^{r,A_r-1,B_r}\\ &\geq \frac{d}{(1+\epsilon)^{i+1}} - \left(\frac{\epsilon}{d}\right)^9\\ &\geq \frac{d}{(1+\epsilon)^{i+2}}. \end{aligned}\tag{5}$$

The second step comes from Eq. (3)(4).

Consequently, we have

$$\begin{aligned}|N(X_i)| - |N(X_{i-1})| &= |N(Z_j \cup X_{i-1})| - |N(X_{i-1})|\\ &\geq \sum_{w\in Z_j\backslash X_{i-1}} \sum_{\substack{s\in N_w\\ N_s\cap(W\backslash Z_j)=\emptyset}} x_{s,w}^{r,A_r-1,B_r}\\ &\geq \frac{d}{(1+\epsilon)^{i+2}}\left|Z_{j-10\log(\frac{d}{\epsilon})\cdot\frac{1}{\epsilon}}\backslash X_{i-1}\right|\\ &\geq \frac{d}{(1+\epsilon)^{i+3}}\left|Z_j\backslash X_{i-1}\right|\\ &= \frac{d}{(1+\epsilon)^{i+3}}\left|X_i\backslash X_{i-1}\right|. \end{aligned}$$

The third step follows from Eq. (5), the fourth step follows from Eq. (2) We complete the induction here. ◀

Let $X_0 = \emptyset, X_1, \ldots, X_A$ be the sets constructed in Lemma 12. In the following two lemmas, we will compare our solution with the optimal solution using the hierarchy of certificates given by Lemma 12.

▶ **Lemma 13.** *Consider a feasible assignment and suppose the loads on the workers are $L_1, \ldots, L_{n_W}$. Then the sequence $(L_1, \ldots, L_{n_W})$ weakly majorizes the following sequence Γ_1:*

- $|X_1\backslash X_0|$ *copies of* $\frac{d}{(1+\epsilon)^{11}}$
- $|X_2\backslash X_1|$ *copies of* $\frac{d}{(1+\epsilon)^{12}}$

$\vdots$

- $|X_A\backslash X_{A-1}|$ *copies of* $\frac{d}{(1+\epsilon)^{10+A}}$
- *All remaining terms are set to* 0

Proof. Since the last $n_W - |X_A|$ term of Γ_1 is 0, it suffices to prove majorization for the first $n_W - |X_A|$ indices. For any index $1 \le j \le n_W - |X_A|$, suppose $|X_r| < j \le |X_{r+1}|$ holds for some r ($0 \le r \le A-1$). By Lemma 12, we have

$$\sum_{w \in X_{r+1}} L_w \ge \sum_{i=0}^{r} |X_{i+1} \backslash X_i| \cdot \frac{d}{(1+\epsilon)^{i+10}}$$

and

$$\sum_{w \in X_r} L_w \ge \sum_{i=0}^{r-1} |X_{i+1} \backslash X_i| \cdot \frac{d}{(1+\epsilon)^{i+10}}$$

Suppose $G(r,j)$ consists of the largest $j - |X_r|$ workers in $X_{r+1} \backslash X_r$, then we have

$$\begin{aligned}
\sum_{w \in X_r \cup G(r,j)} L_w &\ge \sum_{w \in X_r} L_w + \frac{j - |X_r|}{|X_{r+1}| - X_r} \sum_{w \in X_{r+1} \backslash X_r} L_w \\
&= \frac{j - |X_r|}{|X_{r+1}| - X_r} \sum_{w \in X_{r+1}} L_w + \frac{|X_{r+1}| - j}{|X_{r+1}| - X_r} \sum_{w \in X_r} L_w \\
&\ge \frac{j - |X_r|}{|X_{r+1}| - X_r} \cdot \sum_{i=0}^{r} |X_{i+1} \backslash X_i| \cdot \frac{d}{(1+\epsilon)^{i+10}} \\
&\quad + \frac{|X_{r+1}| - j}{|X_{r+1}| - X_r} \sum_{i=0}^{r-1} |X_{i+1} \backslash X_i| \cdot \frac{d}{(1+\epsilon)^{i+10}} \\
&= \sum_{i=0}^{r-1} |X_{i+1} \backslash X_i| \cdot \frac{d}{(1+\epsilon)^{i+10}} + (j - |X_i|) \cdot \frac{d}{(1+\epsilon)^{r+10}}
\end{aligned}$$

Thus completing the proof. ◀

▶ **Lemma 14.** *Consider the assignment returned by our algorithm and suppose the loads on the workers are $L'_1, L'_2, \ldots, L'_{n_W}$. Then the sequence $(L'_1, \ldots, L'_{n_W})$ is weakly majorized by the following sequence Γ_2*

- $|X_1 \backslash X_0|$ *copies of* $\frac{(1+50\epsilon)d}{(1+\epsilon)^1}$
- $|X_2 \backslash X_1|$ *copies of* $\frac{(1+50\epsilon)d}{(1+\epsilon)^2}$

$\vdots$

- $|X_A \backslash X_{A-1}|$ *copies of* $\frac{(1+50\epsilon)d}{(1+\epsilon)^A}$
- *All remaining terms set to* 0

Proof. For each index $1 \le i \le A$, define W^i to be the set of workers that are fixed after completing i full executions of PROPORTIONAL ALLOCATION and capacity update step. Consider any worker w that satisfies

$$L_w \ge \frac{(1+40\epsilon)d}{(1+\epsilon)^i}.$$

By Lemma 7, the capacity of worker w can be bounded as

$$C_w \ge \frac{(1+20\epsilon)d}{(1+\epsilon)^i}.$$

Now it is clear that this can only happen if the worker w is fixed before round i. In particular, the number of such workers is at most $|W^i|$. Meanwhile, we know that $|X_i| \geq |W^i|$ by Lemma 12. Thus, if we only consider the $|X_A|$ largest terms of the sequence $(L'_1, \dots, L'_{n_W})$, they are entry-wise dominated by the terms of Γ_2.

Since all terms after the $|X_A|^{\text{th}}$ term of Γ_2 are 0, to complete the proof of the weak majorization, it suffices to show that the sum of all of the terms of Γ_2 is at least $L'_1 + \dots + L'_{n_W}$. Let U be the set of workers that are not *fixed* at the end of the execution of the algorithm. For all worker $w \in U$, due to the choice of A, we have

$$L'_w \leq (1+10\epsilon)C_w^{A-1,B} = \frac{(1+10\epsilon)d}{(1+\epsilon)^{A-1}} \leq \frac{\epsilon}{d},$$

and therefore,

$$\sum_{w \in U} L'_w \leq \frac{\epsilon n_W}{d}. \tag{6}$$

Since the number of sources is at least $\frac{n_W}{d}$, we have

$$\sum_{w \in W} L'_w \geq \frac{n_W}{d}. \tag{7}$$

WLOG, we can assume $L'_1 \geq L'_2 \geq \dots \geq L'_{n_W}$. We then have

$$\sum_{i=0}^{A-1} |X_{i+1} \backslash X_i| \cdot \frac{(1+40\epsilon)d}{(1+\epsilon)^i} \geq \sum_{j=1}^{|X_A|} L'_j \geq \sum_{w \in W \backslash U} L'_w \geq (1-\epsilon) \sum_{w \in W} L'_w.$$

The second step holds due to the fact that the total number of workers that are fixed by the end of the algorithm is at most $|X_A|$ (see Lemma 12). The last step follows from Eq. (6)(7).

Consequently, we have

$$\sum_{i=0}^{A-1} |X_{i+1} \backslash X_i| \cdot \frac{(1+50\epsilon)d}{(1+\epsilon)^i} \geq \sum_{w \in W} L'_w,$$

which completes the proof. ◀

Combining Lemma 13 and Lemma 14, we get

▶ **Corollary 15.** *Consider the assignment returned by our algorithm, and suppose the loads are $L'_1, \dots, L'_{n_W}$. For any feasible allocation, the loads $L_1, \dots, L_{n_W}$ must satisfy that $(L_1, \dots, L_{n_W})$ weakly majorizes*
$$\left(\frac{L'_1}{1+100\epsilon}, \dots, \frac{L'_{n_W}}{1+100\epsilon}\right)$$

Now we can easily wrap up the proof of Theorem 5.

Proof of Theorem 5. Let the loads in the output of our algorithm be $L'_1, \dots, L'_{n_W}$ and let the loads in the optimal solution be $L_1, \dots, L_{n_W}$. Then

$$\Phi(L'_1, \dots, L'_{n_W}) \leq \Phi\left((1+100\epsilon)L_1, \dots, (1+100\epsilon)L_{n_W}\right)$$

where we use Corollary 15 and Lemma 11. The above inequality immediately implies the desired. ◀

5 Improved algorithm for minimizing max load

If the objective is the max function, i.e., we want to minimize the maximum load on workers, we can further reduce the number of distributed rounds to $O\left(\frac{\log n \log d}{\epsilon^2}\right)$. The idea is to start with a coarse estimation and gradually decrease the precision parameter down to ϵ. The key observation is that each time we decrease the precision value, we only need to run a constant number of rounds of Proportional Allocation to refine our estimation and the number of distributed rounds of the algorithm is dominated by the number of rounds for the smallest precision parameter, which is $O\left(\frac{\log n \log d}{\epsilon^2}\right)$.

In Appendix B, we present an example showing that our analysis of our algorithm is tight. More specifically, we show that any proportional allocation based algorithm that starts from a uniform initialization and has step size bounded by ϵ must run at least $\Omega\left(\frac{\log n \log d}{\epsilon^2}\right)$ rounds to compute a $1+\epsilon$-approximation.

To facilitate the presentation, we first define the precision sequence we use in the algorithm.

▶ **Definition 16.** *We define the sequence $\{\epsilon_r\}_{r\in\mathbb{N}}$ and $\{B_r\}_{r\in\mathbb{N}}$ as follow.*
- *For any $r \geq 0$, $\epsilon_{r+1} = (1+\epsilon_r)^{1/2} - 1$ and $\epsilon_0 = d - 1$.*
- *For any $r \geq 0$, $B_r = 2\log_{1+\epsilon_r}(n)\log_{1+\epsilon_r}(d/\epsilon) + 8\log_{1+\epsilon_r}(n)$*

We further define $R(n,\epsilon)$ to be the smallest index such that $(1+\epsilon_r)^5 \leq 1+\epsilon$.

Algorithm 3 Load Balancing for the Max objective.

```
1:  Initialize weights $a_w = 1$ for all workers $w \in W$.
2:  Initialize capacity upper bounds $C = d_{\max}$.
3:  Set $R = R(n,\epsilon)$.
4:  for $r = 0, 1, \ldots, R-1$ do
5:      Reset $a_w = 1$, $\forall w \in W$
6:      while True do
7:          Run Proportional Allocation for $B_r$ rounds with initial parameters $a_w$, $\epsilon_r$,
    $C_w = C\ \forall w$,
8:          if $\exists w \in W$ such that $L_w \geq (1+\epsilon_r)^4 C$ then
9:              Break the while loop
10:         else
11:             $C \leftarrow \frac{C}{1+\epsilon_r}$
12:     $C \leftarrow C(1+\epsilon_r)^8$
13: for $s, w$ do
14:     Output $x_{s,w} = \frac{a_w}{\sum_{w\in N_s} a_w}$
```

▶ **Theorem 17.** *When the objective is the max function, Algorithm 3 returns an $(1+\epsilon)$ approximate solution to the load balancing problem after $O\left(\frac{\log(n)\log(d/\epsilon)}{\epsilon^2}\right)$ iterations.*

Notations

For any $0 \leq r \leq R-1$, we use A_r to denote the number of calls we make to Proportional Allocation within the while loop of round r. For the load variable L_w, we slightly abuse of notation and for any $0 \leq r \leq R-1, 0 \leq a \leq A_r - 1, 0 \leq t \leq B_r$, we use $L_w^{r,a,t}$ to denote the load on worker w when the algorithm is executed to the timestep r, a, t, i.e. we finished r outer loops and completed a full iterations of Proportional Allocation, and then t rounds within the next iteration of Proportional Allocation. We apply the same notation for a_w and $x_{s,w}$. For the capacity variable, since we maintain the same capacity over all workers and the capacity does not change during the execution of Proportional Allocation, we simplify the notation and use $C^{r,a}$ to denote the capacity on the worker side right after we finished r outer loops and completed a full iterations of Proportional Allocation *but before we perform the update on the capacity.*

The following observation, a restatement of Lemma 6, still holds.

▶ **Lemma 18.** *For any* $0 \le r \le R-1, 0 \le a \le A_r - 1, 0 \le t \le B_r - 1$

- *If* $L_w^{r,a,t} > C^{r,a}$ *then* $\frac{L_w^{r,a,t}}{(1+\epsilon_r)^2} \le L_w^{r,a,t+1} \le L_w^{r,a,t}$
- *If* $L_w^{r,a,t} < C^{r,a}$ *then* $L_w^{r,a,t} \le L_w^{r,a,t+1} \le (1+\epsilon_r)^2 L_w^{r,a,t}$

The following Lemma follows immediately from Lemma 18.

▶ **Lemma 19.** *Consider a worker w such that for some* $0 \le r < R, 0 \le a < A_r$,

$$L_w^{r,a,B_r} \le \frac{C_w^{r,a}}{(1+\epsilon_r)^2}$$

then

$$a_w = (1+\epsilon_r)^{(a+1)B_r}$$

i.e. if a worker is significantly underallocated then its weight must have been increased at every step.

We proceed next to the following key lemma, which says when we break the while loop for any round r, we actually find a certificate lower bound on the optimal value. The proof of Lemma 20 bares some similarities with Lemma 12, the difference is that we need to deal with the case that ϵ_r is large. The detailed proof is provided in Appendix A for completeness.

▶ **Lemma 20.** *For any $0 \le r < R$, we can find a subset of worker $W_r \subseteq W$ such that*

$$|N(W_r)| \ge |W_r| \cdot \frac{C^{r,A_r}}{(1+\epsilon_r)^4}.$$

We next show that at the end of each round of Proportional Allocation, the load on any worker can never significantly exceed its target capacity.

▶ **Lemma 21.** *For all workers w, we have at all timesteps r, a with* $0 \le r < R, 0 \le a \le A_r - 1$

$$L_w^{r,a,B_r} \le (1+\epsilon_r)^5 C^{r,a+1}.$$

Proof. We prove by induction on (r, a). The base case hold trivially as $L_w^{0,0,B_r} \le d_{\max} = C^{0,1}(1+\epsilon_0)$. Suppose the claim holds up to (r, a), it is easy to see it holds for $(r, a+1)$. This is due to Lemma 18 and the fact that we stop if $L_w \ge (1+\epsilon_r)^4 C$ for any worker.

It remains to show that if the claim holds up to $(r, A_r - 1)$, then it also holds for $(r+1, 0)$. This would complete the proof. We prove by contradiction. Suppose $L_w^{r+1,0,B_r} \ge (1+\epsilon_r)^5 C^{r+1,1}$, then we know that we need to break out the loop and followed by Lemma 20, there exists a subset of workers $W_{r+1} \subseteq W$ such that

$$|N(W_{r+1})| \ge |W_{r+1}| \cdot \frac{C^{r+1,1}}{(1+\epsilon_{r+1})^4}$$

This actually says that the optimal solution is at least

$$\mathsf{OPT} \geq \frac{C^{r+1,1}}{(1+\epsilon_{r+1})^4} = \frac{C^{r+1,1}}{(1+\epsilon_r)^2} = C^{r,A_r}(1+\epsilon_r)^6.$$

However, by induction, we have that $L_w^{r,A_r-1,B_r} \leq (1+\epsilon_r)^5 C^{r,A_r}$. this says that we actually have a solution with max load bounded by $(1+\epsilon_r)^5 C^{r,A_r}$, i.e., $\mathsf{OPT} \leq (1+\epsilon_r)^5 C^{r,A_r}$ This comes to a contradiction and we conclude the proof here. ◀

▶ **Lemma 22.** *For $0 \leq r < R$, the optimal solution satisfies*

$$OPT \in [(1+\epsilon_r)^{-4} C^{r,A_r}, (1+\epsilon_r)^5 C^{r,A_r}]$$

Proof. The lower bound comes from Lemma 20. The upper bound comes from Lemma 21. ◀

We can now wrap up the proof of Theorem 17

Proof of Theorem 17. The correctness of the algorithm comes as a direct corollary of Lemma 22. For the running time, for any $0 \leq r < R$, we claim that $A_r \leq 30$. The claim holds trivially for $r = 0$ and for $r \geq 1$, we have

$$\begin{aligned}\mathsf{OPT} &< (1+\epsilon_r)^6 C^{r,A_r} = (1+\epsilon_r)^{6-A_r} C^{r,0} = (1+\epsilon_{r-1})^{3-A_r/2} C^{r,0} \\ &= (1+\epsilon_{r-1})^{11-A_r/2} C^{r-1,A_{r-1}} \leq (1+\epsilon_{r-1})^{15-A_r/2} \mathsf{OPT}\end{aligned}$$

The first inequality follows from the upper bound of Lemma 21, the second step follows from $C^{r,A_r} = (1+\epsilon)^{-A_r} C^{r,0}$. The third step follows from the fact that $(1+\epsilon_r) = (1+\epsilon_{r-1})^{1/2}$, the fourth step follows from the fact that $C^{r,0} = (1+\epsilon_{r-1})^8 C^{r-1,A_{r-1}}$, the last inequality follows from the lower bound of Lemma 21.

The total number of iterations is then bounded by

$$\begin{aligned}\sum_{r=0}^{R} A_r B_r \leq 30 \sum_{r=0}^{R} B_r &\lesssim \sum_{r=0}^{R} \log_{1+\epsilon_r}(n) \log_{1+\epsilon_r}(d/\epsilon_r) + \log_{1+\epsilon_r}(n) \\ &= \sum_{r=0}^{\epsilon_r \geq 1} \log_{1+\epsilon_r}(n) \log_{1+\epsilon_r}(d/\epsilon_r) + \sum_{r:\epsilon_r \leq 1}^{R} \log_{1+\epsilon_r}(n) \log_{1+\epsilon_r}(d/\epsilon_r) \\ &\lesssim \sum_{r=0}^{\epsilon_r \geq 1} \log_{1+\epsilon_r}(n) \log_{1+\epsilon_r}(d/\epsilon_r) + \sum_{r:\epsilon_r \leq 1}^{R} \frac{\log n \log(d/\epsilon)}{\epsilon_r^2} \\ &\lesssim \log n \log d + \frac{\log n \log(d/\epsilon)}{\epsilon^2} \\ &= O\left(\frac{\log n \log(d/\epsilon)}{\epsilon^2}\right)\end{aligned}$$

Thus completing the proof. ◀

References

1 Shipra Agrawal, Vahab Mirrokni, and Morteza Zadimoghaddam. Proportional allocation: Simple, distributed, and diverse matching with high entropy. In *International Conference on Machine Learning*, pages 99–108, 2018.

2 Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1439–1456. SIAM, 2014.

3 Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly linear-time packing and covering lp solvers. *Mathematical Programming*, 175(1-2):307–353, 2019.

4 Sepehr Assadi, Aaron Bernstein, and Zachary Langley. Improved bounds for distributed load balancing. *arXiv preprint*, 2020. `arXiv:2008.04148`.

5 Baruch Awerbuch and Rohit Khandekar. Stateless distributed gradient descent for positive linear programs. *SIAM Journal on Computing*, 38(6):2468–2486, 2009.

6 Petra Berenbrink, Tom Friedetzky, and Zengjian Hu. A new analytical method for parallel, diffusion-type load balancing. *Journal of Parallel and Distributed Computing*, 69(1):54–61, 2009.

7 Petra Berenbrink, Tom Friedetzky, and Russell Martin. Dynamic diffusion load balancing. In *International Colloquium on Automata, Languages, and Programming*, pages 1386–1398. Springer, 2005.

8 Andrzej Czygrinow, Michal Hanćkowiak, Edyta Szymańska, and Wojciech Wawrzyniak. Distributed 2-approximation algorithm for the semi-matching problem. In *International Symposium on Distributed Computing*, pages 210–222. Springer, 2012.

9 Magnús M Halldórsson, Sven Köhler, Boaz Patt-Shamir, and Dror Rawitz. Distributed backup placement in networks. *Distributed Computing*, 31(2):83–98, 2018.

10 Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 448–457, 1993.

11 Michael W Mahoney, Satish Rao, Di Wang, and Peng Zhang. Approximating the solution to mixed packing and covering lps in parallel o (epsilon^{-3}) time. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

12 Albert W Marshall and Frank Proschan. An inequality for convex functions involving majorization. Technical report, BOEING SCIENTIFIC RESEARCH LABS SEATTLE WASH, 1964.

13 Yuval Rabani, Alistair Sinclair, and Rolf Wanka. Local divergence of markov chains and the analysis of iterative load-balancing schemes. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 694–703. IEEE, 1998.

14 Google Search Statistics. *Internet Live Stats*, 2020. URL: `https://www.internetlivestats.com/google-search-statistics/`.

15 Raghu Subramanian and Isaac D Scherson. An analysis of diffusive load-balancing. In *Proceedings of the sixth annual ACM symposium on Parallel algorithms and architectures*, pages 220–225, 1994.

16 Di Wang. *Fast Approximation Algorithms for Positive Linear Programs*. PhD thesis, UC Berkeley, 2017.

17 Di Wang, Satish Rao, and Michael W Mahoney. Unified acceleration method for packing and covering problems via diameter reduction. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

18 Neal E Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings 42nd IEEE symposium on foundations of computer science*, pages 538–546. IEEE, 2001.

A Missing proof from Section 5

Proof of Lemma 20. We prove the lemma for a fixed r. Define

$$F := \left\{w | w \in W, L_w^{r,A_r-1,B_r} > (1+\epsilon_r)^4 C^{r,A_r}\right\},$$

i.e., F is the subset of workers whose load significantly outweigh the target capacity and break the while loop. For any worker $w \in F$, we assert that its weight a_w must decrease every timestep during the last call to PROPORTIONAL ALLOCATION, i.e.,

$$a_w^{r,A_r-1,t+1} = \frac{a_w^{r,A_r-1,t}}{1+\epsilon_r} \quad \forall 0 \le t < B_r.$$

This is because if $L^{r,A_r-1,t_0} < C^{r,A_r-1}$ for some t_0, then by Lemma 18, we know that $L_w^{r,A_r-1,t} \le (1+\epsilon_r)^2 C^{r,A_r-1}$ holds for all $t \ge t_0$, which contradicts with the fact that $L_w^{r,A_r-1,B_r} > (1+\epsilon_r)^4 C^{r,A_r}$. Thus we conclude that

$$a_w^{A_r-1,B_r} \le (1+\epsilon_r)^{(A_r-1)B_r} \quad \forall w \in F.$$

For any $0 \le j \le B_r$, define

$$Z_j = \left\{w | w \in W, a_w^{A_r-1,B_r} \le (1+\epsilon_r)^{(A_r-1)B_r+j}\right\}.$$

We note that $F \subseteq Z_0 \subseteq Z_1 \cdots \subseteq Z_{B_j} = W$ and therefore $Z_0 \ne \emptyset$.

Since $B_r = 2\log_{1+\epsilon_r}(n)\log_{1+\epsilon_r}(d/\epsilon) + 8\log_{1+\epsilon_r}(n)$, there must exist some index j satisfies

$$2\log_{1+\epsilon_r}(d/\epsilon) + 8 \le j < B_r$$

and

$$|Z_j| \le (1+\epsilon_r)|Z_{j-2\log_{1+\epsilon_r}(d/\epsilon)-8}|. \tag{8}$$

We set $W_r = Z_j$ and will show that $|N(Z_j)| \ge |Z_j| \cdot C^{r,A_r}/(1+\epsilon_r)^4$ in the rest of the proof. For any worker $w \in Z_{j-2\log_{1+\epsilon_r}(d/\epsilon)-8}$, if worker w is connected to some source s such that s has another neighbor w' with $w' \notin Z_j$. Then we have

$$\begin{aligned} x_{s,w}^{r,A_r-1,B_r} &\le \frac{a_w^{r,A_r-1,B_r}}{a_w^{r,A_r-1,B_r} + a_{w'}^{r,A_r-1,B_r}} \le \frac{(1+\epsilon_r)^{(A_r-1)B_r+j-2\log_{1+\epsilon_r}(d/\epsilon)-8}}{(1+\epsilon_r)^{(A_r-1)B_r+j+1}} \\ &\le \frac{\epsilon_r^2}{d^2(1+\epsilon_r)^9} \le \frac{\epsilon_r}{d^2(1+\epsilon_r)^8}. \end{aligned}$$

Hence, the total contributions $x_{s,w}^{r,A_r-1,B_r}$ from all sources s with a neighbor not in Z_j to worker w is at most

$$\sum_{\substack{s \in N_w \\ N_s \cap (W \setminus Z_j) \ne \emptyset}} x_{s,w}^{r,A_r-1,B_r} \le \frac{\epsilon_r}{d^2(1+\epsilon_r)^8} \cdot d \le \frac{\epsilon_r}{d(1+\epsilon_r)^8}. \tag{9}$$

Consider the restricted graph between the source set $N(Z_j)$ and the worker set Z_j, for each worker $w \in Z_{j-2\log_{1+\epsilon_r}(d/\epsilon)-8}$, by Lemma 19, we know that its load satisfies

$$L_w^{r,A_r-1,B_r} \ge \frac{C^{r,A_r-1}}{(1+\epsilon_r)^2}.$$

Hence the load of worker w in the restricted graph is at least

$$\begin{aligned}\sum_{\substack{s\in N_w\\ N_s\cap(W\setminus Z_j)=\emptyset}} x_{s,w}^{r,A_r-1,B_r} &= L_w^{r,A_r-1,B_r} - \sum_{\substack{s\in N_w\\ N_s\cap(W\setminus Z_j)\neq\emptyset}} x_{s,w}^{r,A_r-1,B_r}\\ &\geq \frac{C^{r,A_r}}{(1+\epsilon_r)^2} - \frac{\epsilon_r}{d_{\max}(1+\epsilon_r)^8}\\ &\geq \frac{C^{r,A_r}}{(1+\epsilon_r)^3} \qquad (10)\end{aligned}$$

The second step follows from Eq. (9), the last step follows from $C^{r,A_r} \geq (1+\epsilon_r)^{-5}$ holds all the time.

Thus, we have

$$\begin{aligned}|N(Z_j)| &= \sum_{w\in Z_j} \sum_{\substack{s\in N_w\\ N_s\cap(W\setminus Z_j)=\emptyset}} x_{s,w}^{r,A_r-1,B_r}\\ &\geq \sum_{w\in Z_{j-2\log_{1+\epsilon_r}(d_{\max}/\epsilon)-4}} \sum_{\substack{s\in N_w\\ N_s\cap(W\setminus Z_j)=\emptyset}} x_{s,w}^{r,A_r-1,B_r}\\ &\geq |Z_{j-2\log_{1+\epsilon_r}(d_{\max}/\epsilon)-4}| \cdot \frac{C^{r,A_r}}{(1+\epsilon_r)^3}\\ &\geq \frac{C^{r,A_r}}{(1+\epsilon_r)^4} \cdot |Z_j|.\end{aligned}$$

The third step follows from Eq. (10), the last step follows from Eq. (9). We complete the proof here. ◀

B Tightness

Here we present an example showing that our analysis of our algorithm is tight for the case of the max function even if we know the optimum value in advance. More formally, we show that starting from a state where all of the weights are initialized to 1 and the precision parameter is set to ϵ, running $\Omega\left(\frac{\log^2 n}{\epsilon^2}\right)$ iterations of proportional allocation is actually necessary.

Note here we assume $1/\epsilon = n^{o(1)}$ i.e. the desired tolerance is much larger than n^{-1}.

Consider the following graph G. Let $k = 0.5\log n/\epsilon$. We have sets $S_1, S_2, \ldots, S_k$ of sources and sets $W_1, W_2, \ldots, W_k$ of workers. For all i let

$$|S_i| = |W_i| = (1-\epsilon)^i \epsilon n.$$

In G, there is a perfect matching between the vertices of W_i and the vertices of S_i for all $1 \leq i \leq k$. Also, there is a complete bipartite graph between the vertices of S_i and W_{i+1} for $1 \leq i \leq k-1$.

Consider the initial state where all of the sources distribute their load uniformly among the adjacent workers. In this case the maximum load among all of the workers is

$$1 + \frac{|S_{k-1}| \cdot |W_k|}{|W_k|+1} \geq 2.$$

However, in the optimal allocation, each set of sources S_i assigns all of its load to the workers in W_i according to the perfect matching and the maximum load among all of the workers is 1. Now we will show that for some positive constant c, starting from the initial state where all sources distribute their load uniformly, after $\frac{c \log^2 n}{\epsilon^2}$ iterations of proportional allocation, the maximum load among all of the workers is at least $1 + 0.1\epsilon$.

First observe that within each set W_i, by symmetry, all of the weights of the workers are equal at all times. Let w_i^t denote the weight of a worker in W_i after t rounds. If there exists an i such that

$$w_i^t \geq w_{i-1}^t n^{-0.1}$$

then consider the sources in S_{i-1}. The amount of their load going to the workers in W_i is at least $\frac{|S_{i-1}| \cdot |W_i|}{|W_i| + n^{0.1}} \geq |S_i|$. Thus the total load going to workers in $W_k \cup W_{k-1} \cup \cdots \cup W_i$ is at least

$$|S_k| + \cdots + |S_i| + |S_i|.$$

Note that $|S_k|, |S_{k-1}|, \dots, |S_i|$ is a geometric series with ratio $\frac{1}{1-\epsilon}$ so

$$|S_i| \geq 0.1\epsilon(|S_k| + \cdots + |S_i|).$$

Thus, there is some worker among $W_k \cup W_{k-1} \cup \cdots \cup W_i$ with load at least

$$\frac{(1 + 0.1\epsilon)(|S_k| + \cdots + |S_i|)}{(|W_k| + \cdots + |W_i|)} = 1 + 0.1\epsilon.$$

On the other hand, if we have

$$w_i^t \leq w_{i-1}^t n^{-0.1}$$

for all i, then the ratio between the smallest and largest weight among all of the workers must be at least $n^{0.1k}$. However, this ratio is 1 at the beginning and increases by a factor of at most $(1+\epsilon)^2$ each round so the number of rounds must be at least

$$\Omega\left(\frac{\log n}{\epsilon} k\right) = \Omega\left(\frac{\log^2 n}{\epsilon^2}\right).$$

▶ Remark 23. Based on this example, an obvious modification would be to adaptively increase the step size of the proportional allocation updates for certain workers depending on their current load. However, note that initially all workers that are not in W_1 or W_k have load very close to 1. In other words, for every worker, there is another worker in its two-hop neighborhood whose load is very close to the optimal load of 1. If we significantly increased the step size, then we would no longer have the stability conditions (Lemma 7 and Lemma 21) and our analysis would have to be quite different.