Approximate Majorization and Fair Online Load Balancing *

Ashish Goel † Adam Meyerson ‡ Serge Plotkin §

Abstract

This paper relates the notion of fairness in online routing and load balancing to *vector majorization* as developed by Hardy, Littlewood, and Polya. We define α-*supermajorization* as an approximate form of vector majorization, and show that this definition generalizes and strengthens the prefix measure proposed by Kleinberg, Rabani and Tardos as well as the popular notion of *max-min fairness*.

The paper revisits the problem of online load-balancing for unrelated 1-∞ machines from the viewpoint of fairness. We prove that a greedy approach is $O(\log n)$-supermajorized by all other allocations, where n is the number of jobs. This means the greedy approach is globally $O(\log n)$-*fair*. This may be contrasted with polynomial lower bounds presented by Goel, Meyerson, and Plotkin for fair online routing.

We also define a machine-centric view of fairness using the related concept of *submajorization*. We prove that the greedy online algorithm is globally $O(\log m)$-*balanced*, where m is the number of machines.

Proposed Running Head: Approximate Fairness in Scheduling.

1 Introduction

Fair Resource Allocation The problems of online allocation of resources in order to minimize load [6, 3] or maximize profit [4] have been well-studied. In a real system, any allocation must also guarantee some form of fairness. Starving one job of resources in order to increase the allocation to another is not always acceptable. Intuitively, if particular jobs demand conflicting resources, those resources should be allocated in an egalitarian manner.

Before addressing this issue, we must formalize our intuitive notion of fairness. Several candidate definitions have been proposed. The "greatest good for the greatest numbers" criterion is equivalent to maximizing the total bandwidth allocated. This criterion optimizes for the overall usage of resources, but may starve a job which requires many resources in favor of several jobs which require few. Another criterion maximizes the bandwidth allocated to the job which receives least bandwidth. For the case of online load balancing, this was considered by [6, 3] and a greedy allocation was proven to be $O(\log m)$ competitive. However, if two jobs do not conflict (demanding

*A preliminary version of these results appeared in the ACM-SIAM Symposium on Discrete Algorithms, 2001.

†Departments of Management Science and Engineering and (by courtesy) Computer Science, Stanford CA 94305. Research supported by an NSF Career award and an Alfred P. Sloan fellowship. Email: ashishg@stanford.edu.

‡Department of Computer Science, University of California, Los Angeles. Email: awm@cs.ucla.edu.

§Department of Computer Science, Stanford University CA 94305. Supported by ARO Grants DAAG55-98-1-0170 and ONR Grant N00014-98-1-0589. Email: plotkin@cs.stanford.edu.

distinct resources), then a guarantee on the minimum assigned bandwidth does not capture our intuition that these jobs should be essentially independent. Yet another criterion is max-min fairness [1, 2, 5]. Max-min fairness makes sense when the assignment of resources is fixed and we need to determine the split of each resource between the jobs requiring it; however, max-min fairness is poorly defined in the scenario where resource assignment must be performed online [7]. Recently, Kleinberg, Rabani, and Tardos [11] proposed a measure of *global fairness* which unifies all the above criteria. We will extend and strengthen their definition by relating it to the concept of *vector majorization* defined by Hardy, Littlewood, and Polya [9].

We compute a vector representing the amount of resources assigned to each request. This *bandwidth vector* is arranged in nondecreasing order of resources. Given two such vectors X and Y, X is majorized by Y ($X \prec Y$) if the sum of the first k terms of X is at least the sum of the first k terms of Y for every k, and the total sums of the vectors are equal. Conceptually, if X is majorized by Y, then assignment X gives more bandwidth to the requests receiving least, and is therefore more fair. We extend majorization by defining α-supermajorization; X is α-supermajorized by Y if (for every k), α times the sum of the first k terms of X is at least equal to the sum of the first k terms of Y. We present several useful theorems regarding supermajorization which follow directly from the work of Hardy, Littlewood, and Polya [9] on majorization.

We consider an allocation of resources X to be α-fair if every other valid allocation Y satisfies $X \prec^{\alpha} Y$ (Y α-supermajorizes X). If there exists an allocation which is 1-fair, then this allocation is identical to the globally fair allocation defined by Kleinberg, Rabani, and Tardos [11] and also satisfies the property of being *max-min fair*. We believe that the concept of approximate supermajorization will find other important applications in fair resource allocation problems.

After presenting our formal definition of fairness, the remainder of the paper is devoted to the "one-infinity" model of online machine scheduling [6, 3] where each job will run on one of a specified subset of machines. The greedy algorithm for this problem is a slight restatement of Graham's rule [8] – assign each job to the least loaded machine capable of processing the job. We prove that this greedy algorithm obtains a globally $O(\log n)$-fair allocation of resources to the jobs. This may be contrasted with the polynomial lower bounds presented for the more difficult routing problem in [7]. Alternately, we can view fairness from the standpoint of the machines. We show that the loads of machines are $O(\log m)$-balanced. This strengthens previous results by showing that online greedy allocations are close to fair throughout, rather than just at the "endpoint" of the minimum-bandwidth job (or maximum-loaded machine).

Lower bounds on the bandwidth assigned to the least-bandwidth job directly translate to lower bounds on global fairness. It follows that any online algorithm must be globally $\Omega(\log m)$-fair as well as globally $\Omega(\log m)$-balanced [6, 3].

Related work: Kleinberg *et al.* [11] addressed the problem of fair resource allocation in the *offline* setting, and gave a polynomial-time algorithm which computes a globally fair solution given all the requests up-front. This solution is also globally 1-fair by our definition. Goel, Meyerson, and Plotkin [7] gave a globally polylogarithmic-fair online algorithm for the more general problem of routing in communication networks under the assumption that the routing algorithm can exercise some control over the bandwidth allocation. Offline versions of the same problem were studied by Kumar and Kleinberg [12].

2 Measuring Fairness by Majorization

For any vector $a =< a_1, a_2, \ldots, a_n >$, denote the i-th smallest component of a by $a_{(i)}$. Suppose we are given two n-dimensional vectors x and y with non-negative components.

Definition 2.1 *Given two n-dimensional vectors x and y, x is said to be majorized by y (y majorizes x) if each of the following is true:*

1. $\sum_{i=1}^{k} x_{(i)} \geq \sum_{i=1}^{k} y_{(i)}$, *for all* $1 \leq k \leq n$
2. $\sum_{i=1}^{n} x_{(i)} = \sum_{i=1}^{k} y_{(i)}$

We use the notation $x \prec y$ to denote the above relation.

In this paper we will only concern ourselves with vectors where each component is non-negative. Notice that majorization is a property of the multisets $\{x_1, x_2, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$. However it is advantageous to think of them as vectors. Majorization was first studied by Hardy, Littlewood, and Polya [9]. They proved that $\sum_i g(x_i) \leq \sum_i g(y_i)$ for all convex functions g iff $x \prec y$. Another interpretation of this relation is that $x \prec y$ if and only if x is a fairer (than y) allocation of a fixed resource. For a detailed exposition of majorization, see [14]. The concept of majorization was first used in the context of fairness by Lorenz [13].

Hardy, Littlewood, and Polya [10] also proved the following:

Theorem 2.1 *$x \prec y$ iff $x = yP$ for a doubly stochastic matrix P.*

Recall that a doubly stochastic matrix is a matrix in which all rows and columns sum to 1. Similarly, a doubly superstochastic matrix has all rows and columns summing to *at least* 1. Hardy *et al.*define x to be *supermajorized* by y iff $x = yP$ for some doubly superstochastic matrix P.

A concept very similar to supermajorization in the context of fair bandwidth allocation was rediscovered (before our work) by Kleinberg, Rabani, and Tardos [11]. A vector of bandwidth allocations x is said to be α-prefix-competitive with another vector y if $\alpha \sum_{i=1}^{k} x_{(i)} \geq \sum_{i=1}^{k} y_{(i)}$. Also, x is said to be coordinate-wise α-competitive to y if $\alpha x_{(i)} \geq y_{(i)}$. They gave an elegant offline algorithm that obtains a bandwidth allocation that is supermajorized by (i.e. is 1-prefix-competitive to) any other allocation. They further present a solution to the single source unsplittable flow problem that is 2-prefix-competitive to any other solution.

Definition 2.2 *A* globally fair solution *is an assignment of jobs to machines which produces a bandwidth allocation that is supermajorized by any other feasible bandwidth allocation.*

Global fairness is a very strong condition, and cannot be achieved in the context of online load balancing. In fact, it is not clear a priori that this property can be achieved even in the offline case – hence, the result of Kleinberg, Rabani, and Tardos is quite striking. To capture approximate fairness, we introduce α-supermajorization which is a slight variation of the notion of supermajorization defined by Hardy, Littlewood, and Polya.

Definition 2.3 *Given two n-dimensional vectors x and y, x is said to be α-supermajorized by y if $\alpha \sum_{i=1}^{k} x_{(i)} \geq \sum_{i=1}^{k} y_{(i)}$ for all $k \leq n$. This is denoted by $x \prec^{\alpha} y$.*

The following properties of supermajorization are immediate from the above definitions:

Theorem 2.2 *1.* $x \prec^{\alpha} y$ *iff* $\alpha x = yP$ *for some doubly superstochastic matrix* P.

2. $x \prec y \Rightarrow x \prec^{1} y$

3. $x \prec^{\alpha} y, y \prec^{\beta} z \Rightarrow x \prec^{\alpha\beta} z$.

Approximate submajorization can be similarly defined.

Definition 2.4 *Given two* n*-dimensional vectors* x *and* y, x *is said to be* α*-submajorized by* y *if* $\sum_{i=1}^{k} x_{(n+1-i)} \leq \alpha \sum_{i=1}^{k} y_{(n+1-i)}$ *for all* $k \leq n$. *This is denoted by* $x \prec_{\alpha} y$.

Definition 2.5 *A vector is said to be globally* α-fair *if it is* α*-supermajorized by any other feasible vector. A vector is said to be globally* α-balanced *if it is* α*-submajorized by any other feasible vector.*

Theorem 2.3 *1.* $x \prec_{\alpha} y$ *iff* $x = \alpha yP$ *for some doubly substochastic matrix* P.

2. $x \prec y \Rightarrow x \prec_{1} y$

3. $x \prec_{\alpha} y, y \prec_{\beta} z \Rightarrow x \prec_{\alpha\beta} z$.

It is worth noting that α-supermajorization does not necessarily imply α-submajorization (or even close). Consider the n-dimensional vectors $x = \langle 1, 1, \ldots, n+1 \rangle$ and $y = \langle 1, 1, \ldots, 1 \rangle$. Clearly, $x \prec^{2} y$. But $x \prec_{\alpha} y$ is not true for any $\alpha < n + 1$.

In the case of load balancing, global α-fairness will be used for a job-centric view of fairness: the k poorest jobs together in a globally α-fair solution must be at least $1/\alpha$ times as rich as the k poorest jobs in any other allocation. Global α-balance will represent a machine-centric view of fairness: the k most loaded machines in a globally α-balanced solution can together have a load that is at most α times the load of the k most loaded machines in any other allocation. Figure 1 illustrates the difference between approximate submajorization and supermajorization.

2.1 Fairness in One-Infinity Load Balancing

We are given a set of m machines. Jobs arrive one at a time online. Each job x specifies a subset F_x of the machines on which it is allowed to run; we call F_x the feasible set for job x. An online algorithm must assign job x to one of the machines in F_x. The total number of jobs is n.

Each job assigned to machine i receives *bandwidth* equal to $1/L_i$ where L_i is the total number of jobs assigned to machine i.

Definition 2.6 *The* bandwidth vector *for an assignment of jobs to machines is composed of the jobs' bandwidths listed in nondecreasing order.*

Definition 2.7 *The* load vector *for an assignment of jobs to machines is composed of the machines' loads* (L_i).

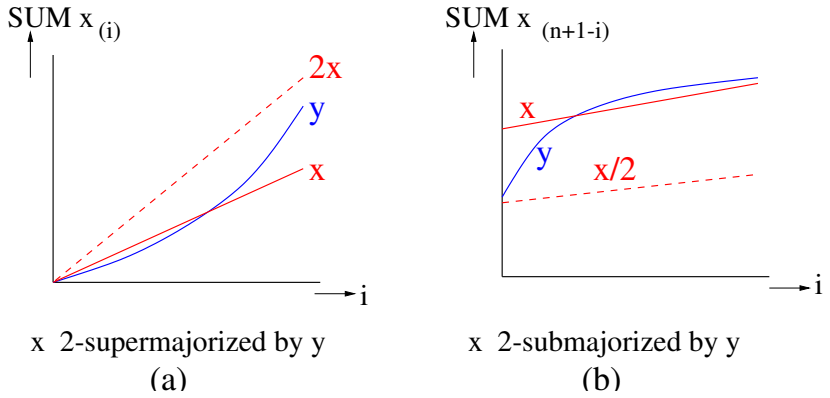

Figure 1: Approximate supermajorization (a) captures fairness when x and y are allocations of resources to jobs, while approximate submajorization (b) captures fairness when x and y are loads on machines. In both cases above, x is at least half as fair/balanced as y.

Definition 2.8 *A solution S is α coordinate-wise competitive if and only if $\alpha S_{(i)} \geq T_{(i)}$ for every coordinate i and every legal solution T.*

We observe that coordinate-wise competitiveness is stronger than supermajorization. In other words, if S is α coordinate-wise competitive, it immediately follows that S is also globally α-fair. Note that the reverse implication does not hold (S may be globally α-fair but not coordinate-wise competitive).

Our algorithm for 1-∞ load balancing will provide a solution which has a globally $O(\log n)$-fair bandwidth vector and a globally $O(\log m)$-balanced load vector.

3 The Greedy Algorithm and its Fairness Properties

We will use a simple greedy algorithm. Whenever a job comes in, we find the least loaded of the machines where the job can run, and assign the job to that machine. Ties are broken arbitrarily. There is no explicit allocation of a specific share of a machine to the jobs assigned to that machine – if a machine has k jobs, we will assume that each job has a bandwidth of $1/k$.

Recall that a 1-balanced allocation exists, as proved by Kleinberg, Rabani, and Tardos [11]. We will refer to a 1-balanced solution as an optimum solution. We will now prove the two major fairness properties of the greedy algorithm: global $O(\log m)$-balance and global $O(\log n)$-fairness.

3.1 Global $O(\log m)$-balance

As mentioned before, the greedy online algorithm is $O(\log m)$ competitive for the maximum load on a machine. We build on this result to relate the load on any machine in the greedy algorithm to the load on the same machine in *any* optimum solution.

Lemma 3.1 *If a machine gets assigned $t > 0$ jobs in an optimum solution, then the greedy algorithm assigns $O(t \log m)$ jobs to that machine.*

Proof: Let L be the vector of loads on the m machines in the optimum solution under consideration.

Let TRANSFER be the relation consisting of all pairs of machines (i, j) such that there exists a job x for which machines i and j are both feasible (i.e., $i, j \in F_x$) but the optimum algorithm allocates it to machine i. Define the relation CHAIN as the reflexive, transitive closure of TRANSFER. Clearly, if $(i, j) \in$ CHAIN then it is possible to modify the optimum allocation such that L_i decreases by exactly one and L_j increases by exactly one. If $L_i - L_j$ were greater than 1 then such a change would lead to a more majorized allocation. Since we start with an optimum solution, there does not exist a pair of machines $(i, j) \in$ CHAIN with $L_i - L_j \geq 2$. Let M_t denote the set of machines

$$\{i : L_i \leq t\}$$

and let M'_t denote the set of machines

$$\{i : \exists j \in M_t \text{ such that} (i, j) \in \text{ CHAIN}\}.$$

By the above argument, the load on any machine in M'_t is at most $t + 1$ in the optimum solution.

Denote by J_t the set of all jobs allocated to some machine in M'_t by the optimum solution. Denote by J^G_t the set of all jobs allocated to some machine in M'_t by the greedy algorithm. Suppose there is a job $x \in J^G_t - J_t$. This job must be allocated to a machine $j \in M'_t$ by the greedy algorithm and to a machine $i \notin M'_t$ by the optimum solution. But then $(i, j) \in$ TRANSFER, which would imply $i \in M'_t$. This is a contradiction, and hence we can conclude that $J^G_t \subseteq J'_t$. This in turn implies that it is possible to schedule all jobs in J^G_t on machines in M'_t such that no machine has load more than $t + 1$. Hence, the maximum load on any machine in M'_t in the greedy allocation is at most $O(t \log m)$. ■

The following theorem follows immediately from Lemma 3.1:

Theorem 3.2 *The greedy online algorithm results in a schedule that is globally $O(\log m)$-balanced.*

In fact, the schedule produced by the greedy algorithm is $O(\log m)$ coordinate-wise competitive to any optimum (i.e., 1-balanced) solution.

3.2 Global $O(\log n)$-fairness

Define the S-load of job x to be the number of earlier jobs allocated to the same machine as x by the greedy algorithm. Define R_q to be the set of jobs with an S-load of at least q. Suppose for some subset R^b_q of R_q, there is a way to assign all the jobs in R^b_q to machines without exceeding b jobs per machine. Then, we have:

Theorem 3.3 *At least $(1/2)|R^b_q|$ jobs have an S-load in the range $[q, q + b)$.*

Proof: Consider the subset X of R^b_q which contains jobs with a S-load of at least $q + b$ or larger. There is an assignment I of jobs in R^b_q to machines without exceeding b jobs per machine. Since the S-load of a job $x \in X$ is at least $q + b$, the machine $I(x)$ must already have a load of at least $q + b$ in the greedy schedule when x arrives. Hence, exactly b jobs assigned to $I(x)$ by the greedy algorithm have an S-load in the range $[q, q + b)$. Since at most b jobs from X are assigned to any one machine by I, there exists a mapping which maps each job in X to a unique job with an S-load in the range $[q, q + b)$. Thus, the number of jobs with an S-load in this range is at least $|X|$.

Now observe that all the jobs in $R_q^b - X$ have an S-load in the range $[q, q+b)$, and hence the number of jobs with an S-load in this range is at least $\max\{|X|, |R_q^b - X|\} \geq (1/2)|R_q^b|$. ■

As mentioned before, there is no explicit bandwidth assignment in our model; a machine is shared equally by all the jobs assigned to that machine. However, for our analysis, it will be convenient to assume that different jobs on the same machine can be assigned different bandwidths, provided only that the total bandwidths of jobs on each machine is at most one. We prove that any globally α-fair solution which can assign arbitrary bandwidth to jobs will remain at least as fair if the jobs split the machines equally instead.

Theorem 3.4 *Suppose there exists an assignment S of bandwidths to jobs and jobs to machines and x is the corresponding vector of bandwidths. Define S' to assign jobs to machines in the same way while splitting bandwidth equally among jobs sent to the same machine, and let x' denote the new vector of bandwidths. Then $x' \prec x$.*

Proof: Let L_i denote the number of jobs on machine i in S. Now define the matrix P as follows:

$$\begin{aligned} P_{j_1,j_2} &= 0 \text{ if } j_1, j_2 \text{ are assigned to different machines in } S \\ &= 1/L_i \text{ otherwise, where } i \text{ is the machine to which } j_1, j_2 \text{ are assigned in } S \end{aligned}$$

If we multiply x with P, each job now gets an average of the bandwidth of all the jobs assigned by S to the same machine. This is precisely x'. Since $x' = xP$ for a doubly stochastic matrix P, we can now invoke Theorem 2.1 to claim $x' \prec x$. ■

3.2.1 Proof of global $O(\log n)$-fairness

Let T be a feasible allocation of bandwidths to jobs. T corresponds to some feasible assignment of jobs to machines, and then some allocation of bandwidths to jobs such that the total bandwidth allocated on any machine does not exceed 1. From T, we obtain T' by using the same assignment of jobs to machines, but now splitting the unit bandwidth of a machine equally among all jobs assigned to it.

By analogy, let S' denote the allocation of bandwidth to jobs obtained by assigning jobs to machines using the greedy algorithm, and then splitting the unit bandwidth of a machine equally among all jobs assigned to it.

A most general result would compare S' to T, and that is what we do. But our analysis takes an indirect route. We first construct another allocation of bandwidths to jobs, S, which agrees with S' in the assignment of jobs to machines, but uses the knowledge of T' to (unevenly) split bandwidth between jobs on the same machine. Since S is used only in the analysis, our assumption that we know the schedule T' in advance is not going to be a problem. We will first show that $S \prec^\alpha T'$ for $\alpha = O(\log n)$ and then invoke Theorem 3.4 twice to claim that $S' \prec^\alpha T$, which is the desired result.

Recall that the S-load of job x is the number of earlier jobs assigned to the same machine as x in the greedy assignment of jobs to machines. Similarly, define the T-load of job x to be the number of earlier jobs scheduled on the same machine in the assignment T (or T', since T and T' agree on the assignment of jobs to machines).

Define t to be the maximum number of jobs which solution T' places on any machine. Clearly, $t \leq n$. Further, we can assume wlog that $m \leq n$ since allowing machines on which T' places zero jobs can only help our algorithm.

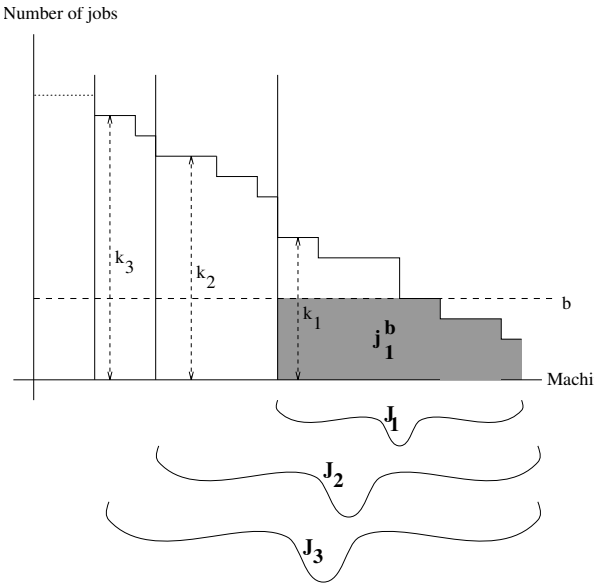

Figure 2: The machine loads for the schedule T'

We define the set J_i (see Figure 2 for an illustration) of machines to consist of all the machines except the $m(1/2)^i$ to which solution T' assigns most jobs. Notice that J_0 is empty and $J_{1+\log m}$ is all the machines. We define k_i to be the maximum number of jobs assigned by T' to a machine in J_i. It follows that $0 = k_0 \leq k_1 \leq k_2 ... \leq k_{1+\log m} = t$.

Define j_i^b to be the number of jobs which solution T' places on machines in J_i, counting at most b jobs per machine (i.e. sum of the min of b and the machine load for each machine). This is equivalent to the number of jobs which have a T-load less than b and are assigned to a machine in J_i by T.

Define r_i^b to be the number of jobs with an S-load less than $b + \Sigma_{x=1}^{i-1} k_x$ (i.e. the sum over machines of the min of the load of the machine in the greedy solution S' and $b + \Sigma_{x=1}^{i-1} k_x$).

Lemma 3.5 *For every $i \geq 1$ and $b \geq 0$ we have $r_i^b \geq j_i^b$.*

Proof: The proof will be by induction on i.

Base Case: $i = 1$. Exactly half the machines are in J_1. We pair each machine in J_1 with a machine not in J_1. The machine not in J_1 always has at least as many jobs as the machine in J_1, because of the definition of the set. Thus the total number of jobs with a T-load less than b is at least $2j_1^b$. Solution T scheduled at least $2j_1^b$ jobs using b capacity per machine, so the greedy algorithm will schedule at least half that many with an S-load less than b (see Theorem 3.3). It follows that $r_1^b \geq j_1^b$ as desired.

Inductive step: If $b > k_i$, then $j_i^b = j_i^{k_i}$ and $r_i^b \geq r_i^{k_i}$. So it suffices to prove the lemma under the assumption that $b \leq k_i$. Since every machine not in J_i has at least k_i jobs on it, there are exactly $mb(1/2)^i$ jobs with a T-load less than b and not assigned to a machine in J_i by T. Hence, it is possible to schedule at least $j_i^b + mb(1/2)^i$ jobs without exceeding b jobs on any machine; in fact T achieves this. By definition, $r_{i-1}^{k_{i-1}}$ is the number of jobs with an S-load

less than $k_1 + k_2 + ... + k_{i-1}$. Thus at least $j_i^b - r_{i-1}^{k_{i-1}} + mb(1/2)^i$ jobs have a T-load less than b but an S-load of at least $k_1 + k_2 + ... + k_{i-1}$.

Informally, in the next b capacity, the greedy algorithm will schedule at least half of them (see Theorem 3.3). Formally, the number of jobs with an S-load in the range $[k_1 + k_2 + ... + k_{i-1}, k_1 + k_2 + ... + k_{i-1} + b)$ is at least half of $j_i^b - r_{i-1}^{k_{i-1}} + mb(1/2)^i$. This means S schedules at least

$$r_i^b \geq r_{i-1}^{k_{i-1}} + \frac{1}{2}\left(j_i^b - r_{i-1}^{k_{i-1}} + \frac{mb}{2^i}\right) \geq \frac{1}{2}\left(j_i^b + r_{i-1}^{k_{i-1}} + \frac{mb}{2^i}\right)$$

jobs with an S-load less than $b + k_1 + k_2 + ... + k_{i-1}$.

Using the inductive hypothesis, we know that $r_{i-1}^{k_{i-1}} \geq j_{i-1}^{k_{i-1}}$.

Each machine in J_{i-1} has at most k_{i-1} jobs on it, so it follows that $j_{i-1}^{k_{i-1}}$ represents all the jobs on machines in J_{i-1}. It follows that $j_{i-1}^{k_{i-1}} \geq j_{i-1}^b$.

There are $m(1/2)^i$ machines in $J_i - J_{i-1}$, and each contributes at most b with a T-load less than b, so we know $j_i^b \leq j_{i-1}^b + mb(1/2)^i$. It follows that $r_{i-1}^{k_{i-1}} + mb(1/2)^i \geq j_i^b$. We can now write

$$r_i^b \geq \frac{1}{2}\left(j_i^b + r_{i-1}^{k_{i-1}} + \frac{mb}{2^i}\right) \geq \frac{1}{2}\left(j_i^b + j_i^b\right) \geq j_i^b.$$

This completes the inductive proof. ■

We will now obtain allocation S from S' by assigning bandwidth to jobs. We define $\alpha = 1 + \log m + \ln t$. Suppose a job is the a-th job on the machine it's running on (i.e. its S-load is $a-1$). There is some unique choice of i and b such that $a = k_1 + k_2 + ... + k_i + b$ and $1 \leq b \leq k_{i+1}$. The job receives bandwidth equal to the minimum of $1/\alpha k_i$ and $1/\alpha b$. If $i = 0$, the bandwidth assigned is $1/\alpha b$. We will first show that this allocation is coordinate-wise competitive, then prove that this is feasible (i.e. no machine has more than one unit of total allocated bandwidth), and finally use Theorem 3.4 to complete the proof of global $O(\log n)$-fairness.

Lemma 3.6 *For any coordinate $j, 1 \leq j \leq n$, we have $\alpha S_{(j)} \geq {T'}_{(j)}$.*

Proof: Let $b = 1/{T'}_{(j)}$. Let i be the maximum integer such that $k_i \leq b$. Any machine not in J_{i+1} has at least $k_{i+1} > b$ jobs on it. Therefore the jobs on this machine receive less than $1/b$ bandwidth in solution T'. It follows that all jobs receiving $1/b$ are on machines with at most b jobs, all of which are in J_{i+1}. Thus at most j_{i+1}^b jobs receive $1/b$ bandwidth or more in solution T'. We conclude that $j \geq n - j_{i+1}^b + 1$.

In S, any job with an S-load less than $k_1 + k_2 + ... + k_i$ will receive at least $1/(\alpha k_i) \geq 1/(\alpha b)$ bandwidth. Jobs with an S-load in the range $[k_1 + k_2 + ... + k_i, k_1 + k_2 + ... + k_i + b)$ receive bandwidth at least equal to $1/\alpha b$. So all jobs which have an S-load less than $k_1 + k_2 + \ldots + k_i + b$ receive at least $1/\alpha b$ bandwidth in S. This means r_{i+1}^b jobs receive at least $1/\alpha b$ bandwidth. Let k be the largest index for which $S_{(k)} < 1/\alpha b$. Then $k \leq n - r_{i+1}^b$.

Lemma 3.5 shows that $r_{i+1}^b \geq j_{i+1}^b$. Hence, $k < j$. It follows that $S_{(j)} \geq 1/\alpha b$, and $\alpha S_{(j)} \geq {T'}_{(j)}$. ■

Coordinate-wise competitiveness immediately implies supermajorization, so we know that $S \prec^\alpha T'$. We still need to show that our bandwidth assignment was legal, i.e., that we did not exceed one unit of bandwidth on any machine.

Lemma 3.7 *The total bandwidth assigned on any machine in the greedy online solution S is at most* 1.

Proof: For $i \geq 1$, let y_i denote the total bandwidth assigned on machine 1 to jobs with an S-load in the range $[k_1 + k_2 + ... + k_i, k_1 + k_2 + ... + k_{i+1})$. Further, let y_0 denote the total bandwidth assigned on machine 1 to jobs with an S-load in the range $[0, k_1)$. Now,

$$y_i = k_i \frac{1}{\alpha k_i} + \frac{1}{\alpha(k_i + 1)} + \frac{1}{\alpha(k_i + 2)} + ... + \frac{1}{\alpha(k_{i+1} - 1)}.$$

We can upper bound the above sum by an integral to obtain

$$y_i \leq \frac{1 + \ln k_{i+1} - \ln k_i}{\alpha}.$$

Similarly, we have

$$y_0 \leq \frac{1}{\alpha}\left(1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{k_1 - 1}\right) \leq \frac{\ln k_1}{\alpha}.$$

The total bandwidth assigned to this machine is $\sum_{i=0}^{1+\log m} y_i$. We sum the upper bounds on y_i's obtained above. This sum telescopes to yield

$$\sum_{i=0}^{1+\log m} y_i \leq \frac{1 + \log m + \ln k_{1+\log m}}{\alpha} = \frac{1 + \log m + \ln t}{\alpha} = 1$$

■

We now proceed to prove our desired result: the greedy algorithm is globally $O(\log n)$-fair.

Theorem 3.8 *Let S' represent the allocation of bandwidth to machines obtained by running the greedy online algorithm and then splitting bandwidth equally on each machine. For any other possible feasible T (including the globally fair assignment), $S' \prec^{\alpha} T$ where $\alpha = 1 + \log m + \ln t$.*

Proof: Lemma 3.6 guarantees that $S \prec^{\alpha} T'$. Lemma 3.7 guarantees that we have a legal bandwidth assignment. We now apply Theorem 3.4 to show that $S' \prec^{1} S$ and $T' \prec^{1} T$. Two applications of Theorem 2.2 suffice to prove $S' \prec^{\alpha} T$ as desired. ■

Finally, we notice that $t \leq n$. We can assume $m \leq n$ since having machines upon which solution T places no jobs can only help our algorithm. It follows that $\alpha \leq 1 + \log n + \ln n$. Thus we have proven global $O(\log n)$ -fairness for the greedy algorithm.

3.3 Lower bounds

Theorem 3.9 *Any online algorithm must be globally $\Omega(\log m)$-fair as well as globally $\Omega(\log m)$-balanced.*

Proof: Consider the quantity L_{max}, the maximum load on a machine. The prefix P_1 of the vector of bandwidths allocated to different jobs is $1/L_{max}$ and the suffix S_1 of the vector of loads on different machines is $L_{\max}$. Thus, if an online algorithm is not β-competitive for L_{max}, it can not be β-fair in terms of the bandwidths allocated to jobs or β-balanced in terms of the loads assigned to machines. A lower-bound of $\Omega(\log m)$ exists on the competitive-ratio of any online algorithm for minimizing L_{max} [6, 3], which proves our theorem. ■

4 An Open Problem

In the job centric model, it would be interesting to either prove global $O(\log m)$-fairness or prove a lower bound of $\Omega(\log n)$, where m is the number of machines and n the number of jobs.

References

[1] Y. Afek, Y. Mansour, and Z. Ostfeld. Convergence complexity of optimistic rate based flow control algorithms. *Journal of Algorithms*, 30(1):106–143, 1999.

[2] Y. Afek, Y. Mansour, and Z. Ostfeld. Phantom: A simple and effective flow control scheme. *Computer Networks*, 32(3):277–305, 2000.

[3] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. *Journal of the ACM*, 44(3):486–504, 1997.

[4] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive online routing. *34th IEEE symposium on Foundations of Computer Science*, pages 32–40, 1993.

[5] B. Awerbuch and Y. Shavitt. Converging to approximated max-min flow fairness in logarithmic time. *Proceedings of the 17th IEEE Infocom conference*, pages 1350–57, 1998.

[6] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignement. *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 203–210, 1992.

[7] A. Goel, A. Meyerson, and S. Plotkin. Combining fairness with throughput: Online routing with multiple objectives. *Journal of Computer and Systems Sciences*, 63(1):62–79, 2001. A preliminary version appeared in ACM Symposium on Theory of Computing, 2000.

[8] R. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. J.*, 45:1563–81, 1966.

[9] G.H. Hardy, J.E. Littlewood, and G. Polya. Some simple inequalities satisfied by convex functions. *Messenger Math.*, 58:145–152, 1929.

[10] G.H. Hardy, J.E. Littlewood, and G. Polya. *Inequalities.* 1st ed., 2nd ed. Cambridge University Press, London and New York., 1934, 1952.

[11] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. *J. Comput. Syst. Sci.*, 63(1):2–20, 2001.

[12] A. Kumar and J. Kleinberg. Fairness measures for resource allocation. *Proceedings of 41st IEEE Symposium on Foundations of Computer Science*, 2000.

[13] M.O. Lorenz. Methods of measuring concentrations of wealth. *J. Amer. Statist. Assoc.*, 9:209–219, 1905.

[14] A.W. Marshall and I. Olkin. *Inequalities: theory of majorization and its applications.* Academic Press (Volume 143 of Mathematics in Science and Engineering), 1979.