Fair Online Load Balancing

Niv Buchbinder
Computer Science Dept., Technion, Haifa, Israel
nivb@cs.technion.ac.il

Joseph (Seffi) Naor*
Microsoft Research, Redmond, WA 98052
naor@cs.technion.ac.il

ABSTRACT

We revisit from a fairness point of view the problem of online load balancing in the restricted assignment model and the 1-∞ model. We consider both a job-centric and a machine-centric view of fairness, as proposed by Goel *et al.* [11]. These notions are equivalent to the approximate notion of prefix competitiveness proposed by Kleinberg, Rabani and Tardos [14], as well as to the notion of approximate majorization, and they generalize the well studied notion of *max-min* fairness.

We resolve a question posed by Goel, Meyerson and Plotkin [11] proving that the greedy strategy is globally $O(\log m)$-fair, where m denotes the number of machines. This result improves upon the analysis of [11] who showed that the greedy strategy is globally $O(\log n)$-fair, where n is the number of jobs. Typically, $n \gg m$, and therefore our improvement is significant. Our proof matches the known lower bound for the problem with respect to the measure of global fairness.

The improved bound is obtained by analyzing, in a more accurate way, the more general restricted assignment model studied previously in [6]. We provide an alternative bound which is not worse than the bounds of [6], and it is strictly better in many cases. The bound we prove is, in fact, much more general and it bounds the load on any prefix of most loaded machines. As a corollary from this more general bound we get that the greedy algorithm results in an assignment that is globally $O(\log m)$-balanced. The last result generalizes the previous result of [11] who proved that the greedy algorithm yields an assignment that is globally $O(\log m)$-balanced for the 1-∞ model.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; G.0 [**Mathematics of Computing**]: General.

General Terms

Algorithms, Theory.

*On leave from the Computer Science Dept., Technion, Haifa, Israel

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SPAA'06, July 30–August 2, 2006, Cambridge, Massachusetts, USA.
Copyright 2006 ACM 1-59593-262-3/06/0007 ...$5.00.

Keywords

Online algorithms, load balancing, scheduling, competitive analysis, greedy algorithm.

1. INTRODUCTION

Load balancing is a fundamental problem in distributed and communication systems and it has been studied extensively in a variety of scenarios and models [4]. Load balancing issues arise, for example, in a multi-processor environment in which tasks, or jobs, need to be assigned to the different processors. The operating system enhances the overall performance of the system by using a balanced allocation of the jobs to the processors. Another example is caching web content in multiple locations to balance the load on web servers, thus achieving fast delivery of content to users. Load balancing problems have been studied in both *offline* and *online* settings. In the offline setting all the jobs are known in advance to the load balancing algorithm, while in the online setting the jobs arrive one-by-one and the load balancing algorithm makes decisions without knowledge of the future.

In this work we revisit the problem of online load balancing in the *restricted assignment* model and in the 1-∞ model. In the restricted assignment model there are m *identical* machines and n jobs. Each job is associated with a non-negative weight w_i and with a subset of feasible machines that may be used to process it. We study an online setting in which jobs arrive one-by-one in an online fashion, and upon arrival of a job we are given its weight as well as a list of feasible machines (out of the m machines) to which it can be assigned. The online algorithm must assign each job to a single machine from its available list and this decision cannot be revoked later on. A special case of the restricted assignment model is the 1-∞ model. The only difference is that in the 1-∞ model all jobs have the same (one unit) weight.

1.1 Fairness

Given a feasible assignment of jobs to machines, the load λ_i on machine i is the sum of the weights of jobs assigned to it. For any job assignment we denote by $\lambda_{(i)}$ the load on the ith most loaded machine, and by $\vec{\lambda} = (\lambda_{(1)}, \lambda_{(2)}, \ldots, \lambda_{(m)})$ the machine load vector that corresponds to the assignment. The quality of an assignment of jobs to machines in a load balancing instance is a function of the load vector. Many quality measures have been suggested. A popular measure is minimizing the maximum load [6, 3], other measures are minimizing the ℓ_2 norm of the machine load vector or any other ℓ_p norm [5].

A different approach to evaluating the quality of an assignment is by considering the load balancing problem from a job-oriented perspective [11]. From a job's point of view the quality of service it gets in an assignment depends on the total weight of jobs that share the resource together with it. Specifically, in the 1-∞ model, if ℓ jobs share the same machine, then each one of them receives a total service/bandwidth of $1/\ell$. From this perspective we may wish to consider the set of bandwidths given to the jobs in the assignment. For each i, $1 \leq i \leq n$, let b_i be the bandwidth given to the ith job and let $b_{(i)}$ be the ith smallest bandwidth taken over all jobs. In order to estimate the quality of an assignment we consider the corresponding bandwidth vector $\vec{B} = (b_{(1)}, b_{(2)}, \ldots, b_{(n)})$ containing the bandwidths given to the jobs sorted in a non-decreasing order. Note that the machine load vector is sorted from the largest value to the smallest value, whereas the bandwidth vector is sorted from the smallest value to the largest value. This notation simplifies our following presentation. One natural quality measure of a solution from a job-oriented perspective is maximizing the bandwidth given to the "poorest" job (i.e. maximizing $b_{(1)}$). This measure relates to the well studied notion of *max-min* fairness [7, 13, 1]. A max-min fair solution is an assignment for which the bandwidth vector is lexicographically maximal. Another quality measure is maximizing the average bandwidth given to the jobs. Other quality measures were also proposed by [16].

In general, it is not always clear how to properly measure the quality of an assignment. Hence it is desirable to find a solution that simultaneously approximates several quality measures [10, 5, 11, 2, 18]. For allocation problems, many of the widely discussed measures of fairness correspond to maximizing a symmetric concave function U, with $U(0) = 0$, that is non-decreasing in each argument [10]. Symmetry corresponds to the assumption that all users are of the same importance. Concavity of the objective function corresponds to the principle of diminishing marginal returns from economics. Note that the constraints of the allocation problem are not necessarily symmetric. Therefore, the optimal solution may not be symmetric even though the objective function is symmetric. The class of such objective functions were called in [10] *canonical utility functions* . In particular, the max-min fairness measure, as well as the average bandwidth objective discussed before, correspond to such canonical utility functions.

Similarly, when considering a congestion problem the goal is, in many cases, minimizing a symmetric convex function C, with $C(0) = 0$, which is non-decreasing in each argument. The class of such functions were referred to in [10] as *canonical congestion functions*. For instance, minimizing the ℓ_p norm of the machine load vector discussed earlier, corresponds to such a canonical congestion function. To capture the notion of simultaneous approximation, Goel et al. [10] gave two formal definitions. We rewrite these definitions in the context of our current load balancing problem.

DEFINITION 1.1. *A bandwidth vector $\vec{B}$ that corresponds to a feasible assignment is a simultaneous α-approximation with respect to all feasible bandwidth vectors if $U(\vec{B}) \geq U(\vec{B}')/\alpha$ for all bandwidths vectors $\vec{B}'$ that correspond to feasible job's assignments and all canonical utility functions U.*

Convex function may be arbitrary steep. Thus, when considering the machine load vector the analogous definition is:

DEFINITION 1.2. *A machine load vector $\vec{\lambda}$ that corresponds to a feasible assignment is a simultaneous α-approximation with respect to all feasible machine load vectors if $C(\frac{1}{\alpha}\vec{\lambda}) \leq C(\vec{\lambda}')$ for all machine load vectors $\vec{\lambda}'$ that correspond to feasible job's assignments and all canonical congestion functions C.*

It is not clear a priori why does there exist an assignment that simultaneously approximates all canonical utility or congestion functions. Also it seems difficult to verify that a certain assignment is a simultaneous approximation since this involves an infinite number of objective functions. To tackle the problem, [10] obtained a different characterization of simultaneous approximation that is more accessible, via the notions of prefix competitiveness and approximate majorization. The notion of *majorization* was first studied in [9, 17]:

DEFINITION 1.3. *Given two n-dimensional vectors x and y, let $x_{(i)}$ and $y_{(i)}$ denote the the ith smallest coordinates in x and y respectively. x is said to be majorized by y (y majorizes x), if:*

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

This relation is denoted by $x \prec y$.

Goel et al. [10] defined two approximate notions of majorization that are equivalent to the notion of prefix competitiveness suggested by Kleinberg et al. [14] as a fairness measure. We restate these definitions in the context of our current load balancing problem.

DEFINITION 1.4. *Given two bandwidth vectors $\vec{B}$ and $\vec{B}'$, $\vec{B}$ is said to be α-supermajorized by $\vec{B}'$ if for all k, $1 \leq k \leq n$, $\sum_{i=1}^{k} b_{(i)} \geq \frac{1}{\alpha} \sum_{i=1}^{k} b'_{(i)}$. This is denoted by $\vec{B} \prec^{\alpha} \vec{B}'$.*

DEFINITION 1.5. *Given two machine load vectors $\vec{\lambda}$ and $\vec{\lambda}'$, $\vec{\lambda}$ is said to be α-submajorized by $\vec{\lambda}'$ if for all k, $1 \leq k \leq n$, $\sum_{i=1}^{k} \lambda_{(i)} \leq \alpha \cdot \sum_{i=1}^{k} \lambda'_{(i)}$. This is denoted by $\vec{\lambda} \prec_{\alpha} \vec{\lambda}'$.*

Note again that the bandwidth vector $\vec{B}$ is sorted from smallest to largest value, while the machine load vector $\vec{\lambda}$ is sorted from largest to smallest value. Based on the work of Hardy, Littlewood, and Polya [9], Goel et al. [10] proved the following two theorems which we restate in the context of our load balancing problem:

THEOREM 1.6 (THM. 2.2 IN [10]). *An assignment results in a bandwidth vector $\vec{B}$ that is that is a simultaneous α-approximation with respect to all feasible bandwidth vectors if and only if the bandwidth vector $\vec{B}$ is α-supermajorized by any feasible bandwidth vector $\vec{B}'$.*

THEOREM 1.7 (THM. 2.3 IN [10]). *An assignment results in a machine load vector $\vec{\lambda}$ that is a simultaneous α-approximation with respect to all feasible machine load vectors if and only if the machine load vector $\vec{\lambda}$ is α-submajorized by any feasible machine load vector $\vec{\lambda}'$.*

Assignments with such properties are referred to as a *globally α-fair* assignment and a *globally α-balanced* assignment:

DEFINITION 1.8. *An assignment that results in a bandwidth vector that is a simultaneous α-approximation with respect to all feasible bandwidth vectors is called a* globally α-fair *assignment.*

DEFINITION 1.9. *An assignment that results in a machine load vector that is a simultaneous α-approximation with respect to all feasible machine load vectors is called a* globally α-balanced *assignment.*

As already stated, it is not at all clear why should a feasible 1-simultaneous approximate solution exist for resource allocation problems. Thus, a first question for any resource allocation problem is whether an α-simultaneous approximate solution exists, for a small value of α. A second (orthogonal) question is whether a simultaneous approximation can be computed efficiently, in both offline and online settings.

Considering the restricted assignment model in the online setting, Goel et al. [11] observed that the lower bound in [6] actually shows that any online algorithm is $\Omega(\log m)$-balanced, where m is the number of machines. For the 1-∞ model the same lower bound shows that any online algorithm is also $\Omega(\log m)$-fair. For the more general restricted assignment model it is not hard to see that no online algorithm can result in an assignment which is globally fair. To see this, consider a simple setting with only two machines and two jobs one of weight 1 and the other of weight $w \gg 1$. The first job that arrives is allowed to be processed on both machines; the next job can only be processed on the machine that the online algorithm selects for the first job. The amount of bandwidth given to the jobs is $1/(w+1)$, and $w/w+1$ respectively. Since there is an assignment such that both jobs gets bandwidth 1, it means that any deterministic online algorithm can only be globally $\Omega(w+1)$-fair.

1.2 Our Results

In this work we revisit the *greedy strategy* for the online restricted assignment model and reanalyze it more accurately. The *greedy strategy* in this setting is the following: upon arrival of a job, the algorithm assigns it to the least loaded machine that can process the job. Ties are broken arbitrarily. Goel *et al.* [11] analyzed the greedy online algorithm in the 1-∞ model from fairness perspective. They showed that the greedy strategy results in an assignment that is globally $O(\log n)$-fair, where n is the number of jobs. This was contrasted with the observation that any online algorithm is globally $\Omega(\log m)$-fair, where m is the number of machines. Typically, $n \gg m$, and thus the gap between the lower and upper bound can be quite large. (In general, there is no reason to assume that m and n are polynomially related.) Closing the gap by tightening either the upper or the lower bound was stated as an open question in [11].

Our first result is improving the upper bound on the greedy online algorithm and closing this gap. We show that the greedy strategy is indeed the best online strategy with respect to global fairness, and it yields an assignment which is globally $O(\log m)$-fair.[1] As stated earlier, achieving online an assignment which is globally $O(\log m)$-fair is only possible for the 1-∞ model. Nevertheless, we obtain our improved upper bound for the 1-∞ model by a finer analysis of the more general restricted assignment model. The greedy algorithm in the restricted assignment model was analyzed in [6] who obtained an upper bound on the maximum load. As a second result we state an interesting alternative upper bound on the maximum load which is always no worse than the bound proved in [6], and it can be strictly better in many cases. The bound we prove is, in fact, much more general and it bounds the load on any prefix of the most loaded machines. As a corollary of this more general bound we get that the greedy algorithm results in an assignment which is globally $O(\log m)$-balanced. The last result generalizes the previous result of [11] who proved that the greedy algorithm yields an assignment which is globally $O(\log m)$-balanced for the 1-∞ model.

1.3 Related Work

Most of the literature on load balancing problems has focused on the machine-centric perspective. From this perspective the most obvious goal is minimizing the maximum load, e.g.,[6, 3]. The restricted assignment model was studied with respect to this measure in [6]. They analyzed the performance of the greedy online strategy, proving that the maximum load is within $O(\log m)$ factor of the optimal load.

Fairness issues in several routing and load balancing models in the offline case were studied by Kleinberg et al. [14]. They defined the notion of prefix competitiveness and a stronger notion of coordinate-wise competitiveness, and considered several offline problems with respect to these measures. Additional offline settings were further studied in [15, 10] who constructed algorithms for finding such approximate solutions, as well as showing lower bounds on the existence of such solutions. The work of Goel et al. [10] made a formal connection between the notion of simultaneous approximation of canonical utility/congestion functions and the notion of approximate majorization and prefix competitiveness. They also designed a general efficient algorithm that finds the best simultaneous approximation solution when the set of constraints is a polynomial-sized linear program.

Offline load balancing in the 1-∞ model was studied in [14] who proved that for this setting there exist an assignment that is 1-globally fair and designed an offline algorithm that efficiently computes such an assignment. The restricted assignment model was studied by Goel et al. [10] and independently by Azar et al. [5]. The motivation in [5] was to simultaneously approximate all ℓ_p norms. They both showed how to efficiently compute (offline) a 2-globally balanced assignment.

The 1-∞ model in an online setting was studied in [11] from fairness perspective. They proved that the greedy strategy is $O(\log n)$-globally fair and $O(\log m)$-globally balanced. The more general case of online routing was considered in [12]. For this setting they showed that there is an $\Omega(m)$ lower bound on the competitive ratio of any online algorithm that splits the bandwidth equally between the jobs. Thus, they considered a more general model in which the online algorithm is allowed to assign weights to the requests that eventually determine the bandwidth given to each request. For this model they designed a poly-logarithmic globally fair algorithm. They also proved, in contrast to the simpler setting of load balancing, that the competitive ratio of any online algorithm in this model is $\Omega(\min\{m, \sqrt{\log n}\})$, and therefore must depend on the number of requests (jobs). The poly-logarithmic upper bound was recently improved by an $O(\log m)$ factor to $O(\log n \log m)$ in [8]. They also showed a lower bound of $\Omega(\log n \log m)$ for a slightly more restrictive model in which the online algorithm should allocate bandwidth directly to the requests.

2. PRELIMINARIES

In the restricted assignment model there are m identical machines, n jobs (where n is not known in advance), and each job is associated with a non-negative weight. The jobs arrive one-by-one in an online fashion, and upon arrival of a job we are given its weight as well as a list of feasible machines (out of the m machines) to which it can be assigned. Thus, the jobs and machines induce a bipartite graph in which there is an edge from job j to machine i if and only if j can be assigned to i. The vertices corresponding to the jobs in the bipartite graph are revealed one-by-one together with the edges adjacent to them. The online algorithm assigns each job to a (single) machine from its available list and this decision cannot be revoked later on. The increase to the load of the machine to which the job is assigned is equal to the weight of the job. We note that the weight of a job is machine independent.

Let $W = (w_1, w_2, \dots, w_n)$ be the non-negative weight vector of the jobs. Let $S(W)$ be a feasible assignment of jobs to machines. The load on machine i, $1 \le i \le m$, denoted by $\lambda_i^{S(W)}$, is the sum of the weights of the jobs that are assigned to it. Let $(\lambda_{(1)}^{S(W)} \ge$

[1] We remark that our results can be stated as the minimum between n and m.

$\lambda_{(2)}^{S(W)} \geq \ldots \geq \lambda_{(m)}^{S(W)})$ be the load vector of assignment $S(W)$, sorted in a non increasing order. We sometimes use the notation $\lambda_{(1)}$ (or λ_1) if the assignment $S(W)$ is obvious from the context.

A job i assigned to a machine with load λ is said to receive service (bandwidth) of $b_i = \frac{w_i}{\lambda}$. For any assignment $S(W)$, let $b_{(i)}$ be the ith smallest bandwidth of a job in the assignment, and let $\vec{B} = (b_{(1)}, b_{(2)}, \ldots, b_{(n)})$ be the bandwidth vector containing the bandwidth assigned to the jobs sorted in a non-decreasing order. Note that the machine load vector is sorted from the largest value to the smallest value, whereas the bandwidth vector is sorted from the smallest value to the largest value. This notation simplifies our following presentation.

3. THE GREEDY ALGORITHM

In this section we analyze the simple greedy load balancing strategy and provide improved and more accurate upper bounds on the load vector generated. We study in this section the more general restricted assignment model in which the jobs have arbitrary nonnegative weight. The *greedy strategy* in this setting is the following: upon arrival of a job, the algorithm assigns it to the least loaded machine that can process the job. Ties are broken arbitrarily. For each assignment $S(W)$ and load b, define

$$N(S(W), b) = \sum_{i=1}^{m} \min\left\{\lambda_{(i)}^{S(W)}, b\right\}$$

I.e., $N(S(W), b)$ is the total weight of the jobs that the assignment $S(W)$ schedules when we truncate the load of each machine at b. Let $G(W)$ be the assignment of the greedy algorithm. For each job i ($1 \leq i \leq n$), let ℓ_i be the load (at the time of assignment) of the machine that i was assigned to by the greedy algorithm. For load b, we define a vector of *residual* weights $W_b = (w_1^b, w_2^b, \ldots, w_n^b)$, as follows:

$$w_i^b = \begin{cases} w_i & \ell_i > b \\ w_i + \ell_i - b & \ell_i \leq b \text{ and } w_i + \ell_i > b \\ 0 & w_i + \ell_i \leq b \end{cases}$$

The residual weight of each job consists of the fraction of the original weight that was scheduled above load b by the greedy algorithm. In the following, we prove a basic lemma generalizing [11, Theorem 3.1]. The lemma states that, for any b, if there is an assignment S of W_b that schedules some weight below load b', then the greedy algorithm would schedule at least half of this weight, with respect to the original weights, between load b and $b + b'$. Formally:

LEMMA 3.1. *Let $S(W_b)$ be an assignment of the jobs with the residual weights for some value of b. Then, for any b':*

$$N(G(W), b' + b) - N(G(W), b) \geq \frac{1}{2} N(S(W_b), b').$$

PROOF. Let T be the set of all residual weights in W_b that were scheduled by $S(W_b)$ on machines up to load b'. The sum of all weights in T is $N(S(W_b), b')$. Each weight belonging to T is of value at most b'. Let $T' \subseteq T$ be the set of weight fractions that the greedy algorithm scheduled above load $b + b'$. Note that if the greedy algorithm scheduled a weight such that only a fraction of it is above $b+b'$ then only this fraction of the original weight belongs to T'. Let $|T|$ and $|T'|$ be the sum of all weights in T and T'.

We first prove that

$$|T'| \leq N(G(W), b' + b) - N(G(W), b).$$

To this end, for each weight in T' we match a corresponding weight that the greedy algorithm schedules between b and $b+b'$. To do so, let $T_m \subseteq T$ be the weight fractions from T that were scheduled in S on some machine m. Let $T_m' \subseteq T_m$ be the weight fractions of T_m that were scheduled by the greedy strategy above load $b + b'$. Let $|T_m|$ and $|T_m'|$ be the sum of all weights in T_m and T_m' respectively. We match the weights in T_m' to a corresponding weights that were scheduled by the greedy algorithm on machine m. We show that the greedy algorithm has enough load on machine m to "pay" for all weights in T_m'.

For each machine m, $|T_m|$ is at most b' since this is the sum of weights S schedules below load b' on m. Thus, if the greedy algorithm schedules on m a total sum of loads of at least $b + b'$, it certainly has enough load between b and $b + b'$ to "pay" for all the matched weight.

Next, assume that the greedy algorithm schedules on machine m a total sum of weights of $b + \ell$ such that $0 \leq \ell < b'$. If $|T_m'| = 0$, we are done. Otherwise, let $w' \in T_m'$ be a strictly positive weight fraction in T_m'. The weight fraction w' is part of an original weight w. The main observation is that this original weight w was scheduled by the greedy algorithm on a machine which had load at most $b + \ell$. This is true since the job can be scheduled on machine m, and the load on machine m, at the time w arrived, is no more than the load on machine m at the end, and the greedy algorithm always prefers the machine with the least load. Also, a positive fraction of the weight w, w', is above $b + b'$. From this observation it follows that the weight fraction of w, which is above b and below $b + b'$, is of value at least $b' - \ell$. This weight fraction belongs to T_m and not to T_m'. Thus,

$$|T_m'| \leq |T_m| - (b' - \ell) \leq b' - (b' - \ell) = \ell,$$

which proves our claim. If the final load on machine m in the greedy assignment is less than b, it means that each job that belongs to T_m was scheduled by the greedy algorithm on a machine with load less than b. If such a job has a fraction that belongs to T_m' (scheduled by the greedy algorithm above $b + b'$), it means that the fraction of it that was scheduled by the greedy between b and $b+b'$ is b'. Thus, it is not possible that S schedules this weight fraction below b'. This means that $|T_m'| = 0$.

The above observations imply on one hand that

$$N(G(W), b' + b) - N(G(W), b) \geq |T'|.$$

On the other hand,

$$N(G(W), b' + b) - N(G(W), b) + |T'| \geq |T| = N(S(W_b), b'),$$

since each weight in T is either scheduled by the greedy between b and $b + b'$ and is therefore counted as part of $N(G(W), b' + b) - N(G(W), b)$, or scheduled above $b + b'$ and, thus, counted as part of $|T'|$. By the above two inequalities we get the desired bound that

$$N(G(W), b' + b) - N(G(W), b) \geq \frac{1}{2} N(S(W_b), b'),$$

which concludes the proof. □

For the next lemma, let $(\lambda_{(1)} \geq \lambda_{(2)} \geq \ldots \geq \lambda_{(m)})$ be the load vector of the greedy assignment sorted in non increasing order. We compare the greedy assignment to an arbitrary assignment S. For the assignment S, let $(\lambda_{(1)}^* \geq \lambda_{(2)}^* \geq \ldots \geq \lambda_{(m)}^*)$ denote the load vector. Note that the identity of the ith most loaded machine may be different in the greedy assignment and in S. Our next lemma shows that the greedy strategy results in a quite balanced assignment. The lemma bounds from below the sum of loads that the greedy assigns below load b, $N(G(W), b)$, for interesting values of b. The bound

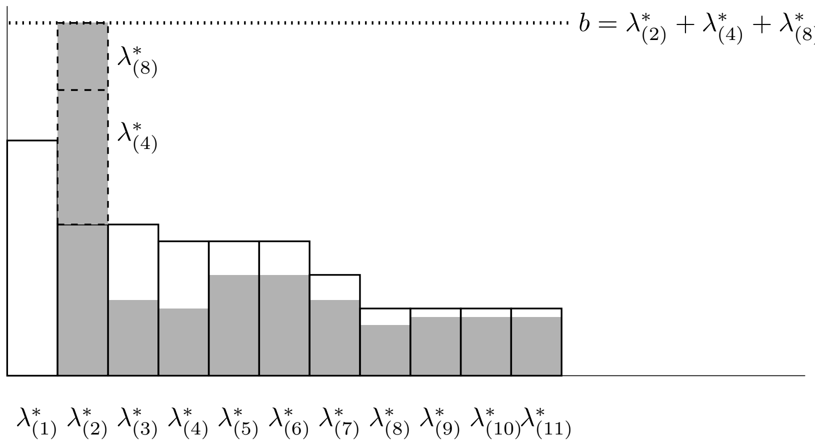


Figure 1: An assignment S and the lower bound for $b = \lambda^*_{(2)} + \lambda^*_{(4)} + \lambda^*_{(8)}$ (by setting $j = 2$ in the lemma). The lower bound corresponds to the gray area: $N(G(W), b) \geq \frac{2}{2}\left(\lambda^*_{(2)} + \lambda^*_{(4)} + \lambda^*_{(8)}\right) + \frac{1}{2}\left(\lambda^*_{(3)} + \lambda^*_{(4)}\right) + \frac{3}{4}\left(\lambda^*_{(5)} + \ldots + \lambda^*_{(8)}\right) + \frac{7}{8}\left(\lambda^*_{(9)} + \ldots + \lambda^*_{(11)}\right)$

is given with respect to the loads $\lambda^*_{(i)}$ in the assignment S. An example of such a lower bound is shown in Figure 1.

LEMMA 3.2. *For any $1 \leq j \leq m$:*

$$N\left(G(W), \sum_{i=0}^{k} \lambda^*_{(j \cdot 2^i)}\right) \geq \frac{j}{2}\left(\sum_{i=0}^{k} \lambda^*_{(j \cdot 2^i)}\right) + \sum_{i=0}^{k} \frac{2^{i+1}-1}{2^{i+1}} \cdot \sum_{r=j\cdot 2^i+1}^{\min\{m, j\cdot 2^{i+1}\}} \lambda^*_{(r)}$$

Where k is the largest value for which $j \cdot 2^k \leq m$.

PROOF. We prove the claim by reverse induction on j. In the following, we sometimes sum over $\lambda^*_{(r)}$ for values $r > m$. For these indices we treat $\lambda^*_{(r)}$ as zero.

Base. For $j > \frac{m}{2}$ the total weight that is scheduled in S, counting at most $\lambda^*_{(j)}$ jobs in each machine is:

$$N(S(W), \lambda^*_{(j)}) = N(S(W_{b=0}), \lambda^*_{(j)}) = j \cdot \lambda^*_{(j)} + \sum_{r=j+1}^{m} \lambda^*_{(r)}$$

where $W_{b=0}$ is the vector of residual weights, setting $b = 0$. This yields precisely the original weight vector. Thus, by Lemma 3.1, the total weight scheduled by the greedy below $\lambda^*_{(j)}$ is at least:

$$N(G(W), \lambda^*_{(j)}) \geq \frac{1}{2} N(S(W_{b=0}), \lambda^*_{(j)}) = \frac{j}{2} \cdot \lambda^*_{(j)} + \frac{1}{2} \sum_{r=j+1}^{m} \lambda^*_{(r)}.$$

Inductive Step. Assume the claim holds for $j > \frac{m}{2^t}$. We prove it for $\frac{m}{2^{t+1}} < j \leq \frac{m}{2^t}$. Let $b = \sum_{i=1}^{k} \lambda^*_{(j \cdot 2^i)} = \sum_{i=0}^{k-1} \lambda^*_{(2j \cdot 2^i)}$. The weight assigned by the greedy algorithm below b is $N(G(W), b)$. Thus,

$$N(S(W_b), \lambda^*_{(j)}) \geq \left(j \cdot \lambda^*_{(j)} + \sum_{r=j+1}^{m} \lambda^*_{(r)}\right) - N(G(W), b).$$

Thus, by lemma 3.1 the total load of jobs that the greedy assignment schedules on machines that have load between b and $b + \lambda^*_{(j)}$ is at least

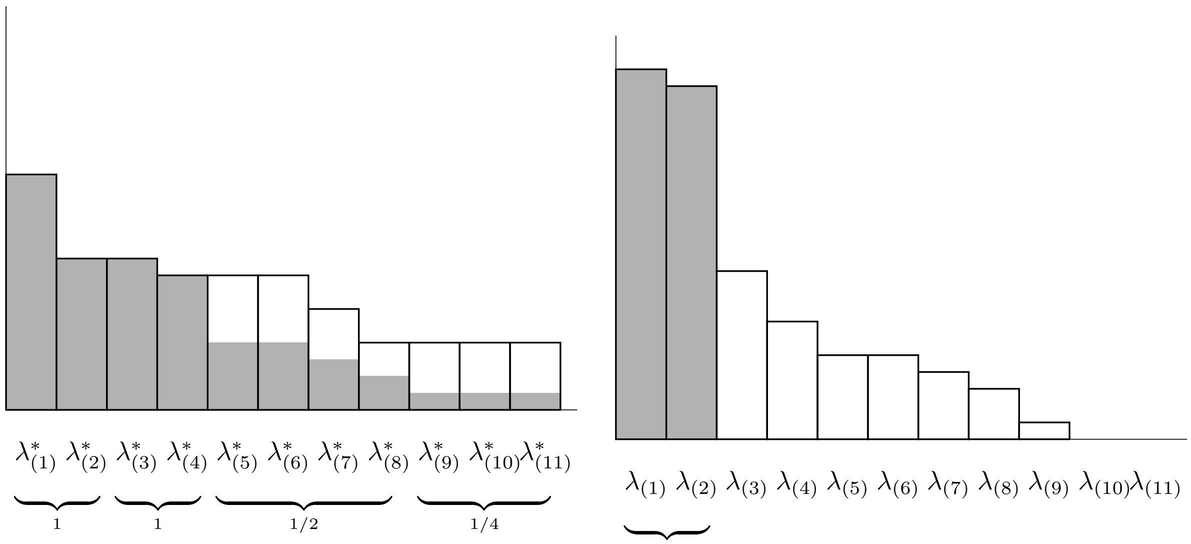
$$\frac{1}{2}\left(j \cdot \lambda^*_{(j)} + \sum_{r=j+1}^{m} \lambda^*_{(r)} - N(G(W), b)\right).$$

On the other hand, by the applying the inductive hypothesis on $2j > \frac{m}{2^t}$, we get that:

$$\begin{aligned} & N[G(W), b] \\ = \; & N\left(G(W), \sum_{i=1}^{k} \lambda^*_{(j \cdot 2^i)}\right) = N\left(G(W), \sum_{i=0}^{k-1} \lambda^*_{(2j \cdot 2^i)}\right) \\ \geq \; & j\left(\sum_{i=0}^{k-1} \lambda^*_{(2j \cdot 2^i)}\right) + \sum_{i=0}^{k-1} \frac{2^{i+1}-1}{2^{i+1}} \sum_{r=2j\cdot 2^i+1}^{\min\{m, 2j\cdot 2^{i+1}\}} \lambda^*_{(r)} \\ = \; & j\left(\sum_{i=1}^{k} \lambda^*_{(j \cdot 2^i)}\right) + \sum_{i=1}^{k} \frac{2^i-1}{2^i} \sum_{r=j\cdot 2^i+1}^{\min\{m, j\cdot 2^{i+1}\}} \lambda^*_{(r)} \end{aligned}$$

Finally, we add the load that the greedy assignment already scheduled below b together with the total load it schedules between $b = \sum_{i=1}^{k} \lambda^*_{(j \cdot 2^i)}$ and $b + \lambda^*_{(j)}$, and get the desired lower bound:

$$\begin{aligned} & N\left(G(W), b + \lambda^*_{(j)}\right) \\ \geq \; & \frac{j \cdot \lambda^*_{(j)} + \sum_{r=j+1}^{m} \lambda^*_{(r)} - N(G(W), b)}{2} + N(G(W), b) && (1) \\ = \; & \frac{j \cdot \lambda^*_{(j)} + \sum_{r=j+1}^{m} \lambda^*_{(r)} + N(G(W), b)}{2} && (2) \\ \geq \; & \frac{j \cdot \lambda^*_{(j)} + \sum_{r=j+1}^{m} \lambda^*_{(r)}}{2} \\ + \; & \frac{1}{2}\left[j\left(\sum_{i=1}^{k} \lambda^*_{(j \cdot 2^i)}\right) + \sum_{i=1}^{k} \frac{2^i-1}{2^i} \sum_{r=j\cdot 2^i+1}^{\min\{m, j\cdot 2^{i+1}\}} \lambda^*_{(r)}\right] && (3) \\ = \; & \frac{j}{2}\left(\sum_{i=0}^{k} \lambda^*_{(j \cdot 2^i)}\right) \\ + \; & \frac{1}{2} \sum_{r=j+1}^{m} \lambda^*_{(r)} + \frac{1}{2} \sum_{i=1}^{k} \frac{2^i-1}{2^i} \sum_{r=j\cdot 2^i+1}^{\min\{m, j\cdot 2^{i+1}\}} \lambda^*_{(r)} && (4) \end{aligned}$$



(a) (b)

Figure 2: An example of an upper bound on the total load on the first two most loaded machines. Applying the lemma for $j = 2$ we get: $\lambda_{(1)} + \lambda_{(2)} \leq \lambda^*_{(1)} + \ldots + \lambda^*_{(4)} + \frac{1}{2}\left(\lambda^*_{(5)} + \ldots + \lambda^*_{(8)}\right) + \frac{1}{4}\left(\lambda^*_{(9)} + \ldots + \lambda^*_{(11)}\right)$

$$= \frac{j}{2}\left(\sum_{i=0}^{k} \lambda^*_{(j\cdot 2^i)}\right) + \sum_{i=0}^{k} \frac{2^{i+1}-1}{2^{i+1}} \sum_{r=j\cdot 2^i+1}^{\min\{m, j\cdot 2^{i+1}\}} \lambda^*_{(r)} \qquad (5)$$

□

Next, we derive an upper bound on the sum of loads of any prefix of machines, ordered in a non-increasing order of loads. As in Lemma 3.2, the bound is stated as a function of the loads $\lambda^*_{(i)}$ in any assignment S against which we compare the greedy assignment. An example of an assignment S is shown in Figure $2(a)$ while Figure $2(b)$ shows a corresponding possible assignment generated by the greedy algorithm online. Setting $j = 2$ in Lemma 3.3 yields a lower bound on the total sum of weights of jobs that the greedy assignment schedules on the two most loaded machines (shown in gray in Figure $2(b)$). The lower bound corresponds to the gray area in Figure $2(a)$. The crucial observation is that the height of the gray area in Figure $2(a)$ drops exponentially.

LEMMA 3.3. *For any $1 \leq j \leq m$:*

$$\sum_{i=1}^{j} \lambda_{(i)} \leq \sum_{i=1}^{j} \lambda^*_{(i)} + \sum_{i=0}^{k} \frac{1}{2^i} \sum_{r=j\cdot 2^i+1}^{\min\{m, 2j\cdot 2^i\}} \lambda^*_{(r)}$$

Where k is the largest value for which $j \cdot 2^k \leq m$.

PROOF. The total sum of weights is $\sum_{i=1}^{m} \lambda^*_{(i)}$. Therefore, for any capacity b we can bound from above the sum of weights of jobs on the j most loaded machines in the greedy assignment by

$$\left[\sum_{i=1}^{m} \lambda^*_{(i)} - N\left(G(W), b\right)\right] + j \cdot b.$$

We set $b = \sum_{i=0}^{k-1} \lambda^*_{(2j\cdot 2^i)}$. Note that the value $k-1$ is indeed feasible for the value $2j$. Using Lemma 3.2 we get the following bound for any $1 \leq j \leq m$:

$$\begin{aligned}
\sum_{i=1}^{j} \lambda_{(i)} &\leq \sum_{i=1}^{m} \lambda^*_{(i)} - N\left(G(W), b\right) + j \cdot \left(\sum_{i=0}^{k-1} \lambda^*_{(2j\cdot 2^i)}\right) \\
&\leq \sum_{i=1}^{m} \lambda^*_i - \frac{2j}{2}\left(\sum_{i=0}^{k-1} \lambda^*_{(2j\cdot 2^i)}\right) \\
&\quad - \sum_{i=0}^{k-1} \frac{2^{i+1}-1}{2^{i+1}} \sum_{r=2j\cdot 2^i+1}^{\min\{m, 2j\cdot 2^{i+1}\}} \lambda^*_{(r)} + j \cdot \left(\sum_{i=0}^{k-1} \lambda^*_{(2j\cdot 2^i)}\right) \\
&= \sum_{i=1}^{m} \lambda^*_i - \sum_{i=0}^{k-1} \frac{2^{i+1}-1}{2^{i+1}} \sum_{r=2j\cdot 2^i+1}^{\min\{m, 2j\cdot 2^{i+1}\}} \lambda^*_{(r)} \\
&= \sum_{i=1}^{j} \lambda^*_{(i)} + \sum_{i=0}^{k} \frac{1}{2^i} \sum_{r=j\cdot 2^i+1}^{\min\{m, 2j\cdot 2^i\}} \lambda^*_{(r)},
\end{aligned}$$

where the second inequality follows from Lemma 3.2. □

Lemma 3.3 yields two almost immediate results. The first result is an alternative bound on the maximum load, stated as a function of any assignment S (and specifically the optimal assignment). This bound never exceeds the bound proved in [6] and in most cases it is strictly tighter. The bounds are only equal in the case in which there is an optimal assignment which is fully balanced, i.e. the loads on the machines are all equal. The second result is that the greedy assignment is $O(\log m)$-balanced. This property was proved in [11] for the simpler 1-∞ model in which all jobs have the same weight.

THEOREM 3.4. *Let k be the largest value for which $2^k \leq m$. Then:*

1. $\lambda_{(1)} \leq \lambda^*_{(1)} + \sum_{i=0}^{k} \frac{1}{2^i} \sum_{r=\cdot 2^i+1}^{\min\{m, \cdot 2^{i+1}\}} \lambda^*_{(r)}$
2. *The greedy assignment is globally $O(\log m)$-balanced.*

PROOF. Part (1) follows immediately from Lemma 3.3 by setting $j = 1$. Part (2) is obtained from the following observation. By Theorem 1.7 we only need to show that the machine load vector $\vec{\lambda}$ that corresponds to the online greedy assignment is $O(\log m)$-submajorized by any feasible machine load vector $\vec{\lambda}^*$. For a given j, $1 \leq j \leq m$, let k be the largest value for which $j \cdot 2^k \leq m$.

Applying Lemma 3.3 we get that:

$$\begin{aligned}\sum_{i=1}^{j}\lambda_{(i)} &\le \sum_{i=1}^{j}\lambda^*_{(i)}+\sum_{i=0}^{k}\frac{1}{2^i}\sum_{r=j\cdot 2^i+1}^{\min\{m,2j\cdot 2^i\}}\lambda^*_{(r)}\\ &\le \sum_{i=1}^{j}\lambda^*_{(i)}+\sum_{i=0}^{k}\sum_{r=j\cdot 2^i+1}^{\min\{m,j\cdot 2^i+j\}}\lambda^*_{(r)}\\ &\le \sum_{i=1}^{j}\lambda^*_{(i)}+\sum_{r=j\cdot 2^i+1}^{\min\{m,j\cdot 2^i+j(k+1)\}}\lambda^*_{(r)}\\ &\le \sum_{i=1}^{j\cdot(k+2)}\lambda^*_{(i)}\le\sum_{i=1}^{j\cdot O(\log m)}\lambda^*_{(i)}\\ &= O(\log m)\cdot\sum_{i=1}^{j}\lambda^*_{(i)}\end{aligned}$$

All inequalities follow since the values $\lambda^*_{(i)}$ are sorted in a non-decreasing order. $\square$

4. GREEDY IS GLOBALLY $O(\log m)$-FAIR

In this section we use the tighter bounds proved in Section 3 to obtain our main result proving that the greedy algorithm is globally $O(\log m)$-fair. As we noted earlier, for the more general restricted assignment model in which the job weights are arbitrary it is not possible to achieve this result. Thus, we study the simpler setting in which all jobs have equal weight (the 1-∞ model), assumed to be 1 without loss of generality. In this setting the load on each machine is simply the number of jobs that are assigned to it. Therefore, all our previous lemmas now reduce to bounds on the total number of jobs on the machines. Before proving our main theorem we need an additional technical lemma regarding the properties of supermajorization.

LEMMA 4.1. *For any two positive n-dimensional vectors x and y, $x \prec^{\alpha} y$ if and only if, for each coordinate j such that $x_{(j+1)} > x_{(j)}$, it holds that $\alpha\cdot\sum_{i=1}^{j}x_{(i)}\ge\sum_{i=1}^{j}y_{(i)}$, and $\alpha\cdot\sum_{i=1}^{n}x_{(i)}\ge\sum_{i=1}^{n}y_{(i)}$.*

PROOF. If $x \prec^{\alpha} y$ then the statement is true for all values of j and so this direction is immediate. For the other direction, assume to the contrary that for each coordinate j such that $x_{(j+1)} > x_{(j)}$ it holds that $\alpha\cdot\sum_{i=1}^{j}x_{(i)}\ge\sum_{i=1}^{j}y_{(i)}$ and also $\alpha\cdot\sum_{i=1}^{n}x_{(i)}\ge\sum_{i=1}^{n}y_{(i)}$, yet $x \prec^{\alpha} y$ does not hold. Let j' be the smallest coordinate for which $\alpha\cdot\sum_{i=1}^{j'}x_{(i)}<\sum_{i=1}^{j'}y_{(i)}$. Since j' is the smallest coordinate for which this happens, then $\alpha\cdot\sum_{i=1}^{j'-1}x_{(i)}\ge\sum_{i=1}^{j'-1}y_{(i)}$. Thus, it must hold that $\alpha\cdot x_{(j')}<y_{(j')}$. Let $j''>j'$ be the first coordinate such that $x_{(j''+1)}>x_{(j'')}$ or $j''=n$ if there is no such coordinate. Thus, we get that:

$$\begin{aligned}\alpha\cdot\sum_{i=1}^{j''}x_{(i)} &= \alpha\cdot\sum_{i=1}^{j'}x_{(i)}+\alpha\cdot\sum_{i=j'+1}^{j''}x_{(i)}\\ &< \sum_{i=1}^{j'}y_{(i)}+\alpha\cdot\sum_{i=j'+1}^{j''}x_{(i)} &(6)\\ &= \sum_{i=1}^{j'}y_{(i)}+\alpha(j''-j')\cdot x_{(i)} &(7)\\ &< \sum_{i=1}^{j'}y_{(j')}+(j''-j')\cdot y_{(j')} &(8)\\ &\le \sum_{i=1}^{j'}y_{(i)}+\sum_{i=j'+1}^{j''}y_{(i)}=\sum_{i=1}^{j''}y_{(i)} &(9)\end{aligned}$$

Inequality 6 follows by our contradiction assumption. Equality 7 follows since j'' is the first coordinate for which $x_{(j''+1)}>x_{(j'')}$ so until this coordinate all coordinates are equal. Inequality 8 follows by our observation that $\alpha\cdot x_{(j')}<y_{(j')}$. Inequality 9 follows since the values of the vector y are sorted in a non-decreasing order. Since j'' is a coordinate such that $x_{(j''+1)}>x_{(j'')}$ or $j''=n$ this contradicts our assumption that $\alpha\cdot\sum_{i=1}^{j''}x_{(i)}\ge\sum_{i=1}^{j}y_{(i)}$. $\square$

Finally, we prove that the bandwidth vector the corresponds to the greedy assignment is globally $O(\log m)$-fair. This is done by proving that this vector is $O(\log m)$-supermajorized by any other feasible bandwidth vector. Let the machine load vector that corresponds to the greedy assignment be $(\lambda_{(1)}\ge\lambda_{(2)}\ge\ldots\ge\lambda_{(m)})$. Then, the corresponding bandwidth vector that is composed of bandwidths given to the n jobs is:

$$\vec{B}=(\underbrace{\frac{1}{\lambda_{(1)}},\frac{1}{\lambda_{(1)}},\ldots,\frac{1}{\lambda_{(1)}}}_{\lambda_{(1)}\text{ times}},\underbrace{\frac{1}{\lambda_{(2)}},\ldots,\frac{1}{\lambda_{(2)}}}_{\lambda_{(2)}\text{ times}},\ldots,\underbrace{\frac{1}{\lambda_{(m)}},\ldots,\frac{1}{\lambda_{(m)}}}_{\lambda_{(m)}\text{ times}})$$

The vector $\vec{B}$ is an n-dimensional vector (coordinate for each job) in which the first $\lambda_{(1)}$ coordinates are $\frac{1}{\lambda_{(1)}}$, the next $\lambda_{(2)}$ coordinates are $\frac{1}{\lambda_{(2)}}$, and so on. Note that it might be that the greedy assignment does not use all the possible machines. In that case, several of the last machines have $\lambda_{(i)}=0$ (the least loaded machines in the assignment). Thus, the last coordinates in the bandwidth vector are actually $\frac{1}{\lambda_{(i)}}$ that corresponds to jobs on the least loaded machine that is still used by the greedy assignment. Let $B_{(i)}$ be the ith smallest coordinate in the vector $\vec{B}$. We compare $\vec{B}$ to the bandwidth vector $\vec{B}'$ of any that corresponds to any other assignment S. This bandwidth vector is of the form:

$$\vec{B}'=(\underbrace{\frac{1}{\lambda^*_{(1)}},\frac{1}{\lambda^*_{(1)}},\ldots,\frac{1}{\lambda^*_{(1)}}}_{\lambda^*_{(1)}\text{ times}},\underbrace{\frac{1}{\lambda^*_{(2)}},\ldots,\frac{1}{\lambda^*_{(2)}}}_{\lambda^*_{(2)}\text{ times}},\ldots,\underbrace{\frac{1}{\lambda^*_{(m)}},\ldots,\frac{1}{\lambda^*_{(m)}}}_{\lambda^*_{(m)}\text{ times}})$$

THEOREM 4.2. *The greedy algorithm results in an assignment which is globally $O(\log m)$-fair.*

PROOF. By Theorem 1.6 we only need to show that the bandwidth vector $\vec{B}$ that corresponds to the online greedy assignment is $O(\log m)$-supermajorized by any feasible bandwidth vector $\vec{B}'$. By Lemma 4.1, we only need to prove that $O(\log m)\cdot\sum_{i=1}^{j}B_{(i)}\ge\sum_{i=1}^{j}B'_{(i)}$ for coordinates j, such that $B_{(j)}<B_{(j+1)}$ and that $\alpha\cdot\sum_{i=1}^{n}B_{(i)}\ge\sum_{i=1}^{n}B'_{(i)}$. This means that the following values of j should be inspected: $\lambda_{(1)},\lambda_{(1)}+\lambda_{(2)},\lambda_{(1)}+\lambda_{(2)}+\lambda_{(3)}$, etc. First, note that we may assume without loss of generality that for any i, $\lambda^*_{(i)}>0$. That is, the assignment that we compare ourselves to uses all the machines. If this is not true we can either change the assignment so it has a strictly better bandwidth vector, or discard some of the machines in the load balancing instance. Using this assumption we get by Lemma 3.3 that:

$$\sum_{i=1}^{m/2-1}\lambda_{(i)}\le\sum_{i=1}^{m-2}\lambda^*_{(i)}+\frac{1}{2}\left(\lambda^*_{(m-1)}+\lambda^*_{(m)}\right)<\sum_{i=1}^{m}\lambda^*_{(i)}$$

This simply tell us that $\lambda_{(m/2)} > 0$ which means that the greedy algorithm uses in its solution at least half of the machines. This means that for values of j for which we sum on the bandwidth of at least $m/2$ machines, the total bandwidth allocated to the jobs is at least $m/2$. Since any assignment can allocate a total bandwidth of at most m, we should not concern ourselves with these prefixes.

For all other values of j we know that the greedy algorithm uses these machines and thus for these values of j, $\sum_{i=1}^{j} B_{(i)} = \ell$, where ℓ is the number of machines that we sum up on. Thus, for each $1 \leq \ell \leq m/2$ and $j = \sum_{i=1}^{\ell} \lambda_{(i)}$, we need to prove that:

$$\sum_{i=1}^{j} B'_{(i)} \leq \ell \cdot O(\log m)$$

In the schedule B', the first $\lambda^*_{(1)}$ jobs get bandwidth $\frac{1}{\lambda^*_{(1)}}$, the next $\lambda^*_{(2)}$ jobs get bandwidth $\frac{1}{\lambda^*_{(2)}}$, and so on. We sum up the bandwidth that the first $\sum_{i=1}^{\ell} \lambda_{(i)}$ jobs get, and to this end we define an indicator χ_t for the event $t \leq \sum_{i=1}^{\ell} \lambda_{(i)}$. Using this notation we get that $\sum_{i=1}^{j} B'_{(i)} = \sum_{p=1}^{m} \sum_{i=1}^{\lambda^*_{(p)}} \frac{\chi_t}{\lambda^*_{(p)}}$, where $t = \sum_{r=1}^{p-1} \lambda^*_{(r)} + i$. Thus, we would like to find out when $\sum_{i=1}^{\ell} \lambda_{(i)} = \sum_{r=1}^{p-1} \lambda^*_{(r)} + i$ happens, since after this point $\chi_t = 0$. If we take k to be the largest value for which $\ell \cdot 2^k \leq m$, then by Lemma 3.3 we get that:

$$\begin{aligned}
\sum_{i=1}^{\ell} \lambda_{(i)} &\leq \sum_{i=1}^{\ell} \lambda^*_{(i)} + \sum_{i=0}^{k} \frac{1}{2^i} \sum_{r=\ell\cdot 2^i+1}^{\min\{m, 2\ell\cdot 2^i\}} \lambda^*_{(r)} \\
&\leq \sum_{i=1}^{\ell} \lambda^*_{(i)} + \sum_{i=0}^{k} \sum_{r=\ell\cdot 2^i+1}^{\min\{m, \ell\cdot 2^i+\ell\}} \lambda^*_{(r)} \\
&\leq \sum_{i=1}^{\ell} \lambda^*_{(i)} + \sum_{r=\ell\cdot 2^i+1}^{\min\{m, \ell\cdot 2^i+\ell(k+1)\}} \lambda^*_{(r)} \\
&\leq \sum_{i=1}^{\ell\cdot(k+2)} \lambda^*_{(i)} \leq \sum_{i=1}^{\ell\cdot O(\log m)} \lambda^*_{(i)}
\end{aligned}$$

Thus, when $p > \ell \cdot c \cdot \log m$, for some constant c, $\chi_t = 0$, and therefore,

$$\begin{aligned}
\sum_{i=1}^{j} B'_{(i)} &= \sum_{j=1}^{m} \sum_{i=1}^{\lambda^*_{(j)}} \frac{\chi_t}{\lambda^*_{(j)}} \leq \sum_{j=1}^{\ell\cdot O(\log m)} \sum_{i=1}^{\lambda^*_{(j)}} \frac{\chi_t}{\lambda^*_{(j)}} \\
&\leq \sum_{j=1}^{\ell\cdot O(\log m)} 1 = \ell \cdot O(\log m)
\end{aligned}$$

concluding the proof. □

5. REFERENCES

[1] Miriam Allalouf and Yuval Shavitt. Maximum flow routing with weighted max-min fairness. In *First International Workshop on QoS Routing (WQoSR)*, 2004.

[2] J. A. Aslam, A. Rasala, C. Stein, and N. Young. Improved bicriteria existence theorems for scheduling. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, pages 846–847, 1999.

[3] James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997.

[4] Yossi Azar. On-line load balancing. *Online Algorithms - The State of the Art, Springer*, (8):178–195, 1998.

[5] Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52(2):120–133, 2004.

[6] Yossi Azar, Joseph Naor, and Raphael Rom. The competitiveness of on-line assignments. *Journal of Algorithms*, 18:221–237, 1995.

[7] Dimitri Bertsekas and Robert Gallager. *Data networks*. Prentice-Hall, Inc., 1987.

[8] N. Buchbinder and J. Naor. A primal-dual approach to online routing and packing. In *Manuscript*, 2006.

[9] John E. Littlewood Godfrey H. Hardy and George P'olya. Some simple inequalities satisfied by convex functions. In *Messenger Math*, pages 58:145–152, 1929.

[10] Ashish Goel and Adam Meyerson. Simultaneous optimization via approximate majorization for concave profits or convex costs. 2005.

[11] Ashish Goel, Adam Meyerson, and Serge A. Plotkin. Approximate majorization and fair online load balancing. In *Symposium on Discrete Algorithms*, pages 384–390, 2001.

[12] Ashish Goel, Adam Meyerson, and Serge A. Plotkin. Combining fairness with throughput: Online routing with multiple objectives. *J. Comput. Syst. Sci.*, 63(1):62–79, 2001.

[13] J.M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, 29(7):954–962, 1981.

[14] Jon Kleinberg, Eva Tardos, and Yuval Rabani. Fairness in routing and load balancing. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 568, 1999.

[15] Amit Kumar and Jon M. Kleinberg. Fairness measures for resource allocation. In *IEEE Symposium on Foundations of Computer Science*, pages 75–85, 2000.

[16] R. K. lain, Dab-Ming Chiu, and William Howe. A quantitative measure of fairness and discrimination for resource allocation in shared systems. In *DEC Res. Rep. TR-301*, 1984.

[17] A. W. Marshal and I. Olkin. Inequalities: Theory of majorization and its applications. In *volume 143 of Mathematics in Science and Engineering*. Academic Press, New York, 1979.

[18] Clifford Stein and Joel Wein. On the existence of schedules that are near-optimal for both makespan and total weighted completion time. Technical Report TR96-295, 1996.